UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria electrònica, industrial i automàtica

ROBOTS PER A APLICACIONS FORESTALS

Memòria i Annexos

Autor: Sergio Ordiaz Bonet
Marc Guarnido Arquero
Director: Joan Domingo Peña

Convocatòria: Octubre de 2019

Resum

El principal objectiu d'aquest projecte és dissenyar un sistema format per tres robots que permetin reforestar terrenys de forma autònoma, on cadascun tindrà una funció concreta però no deixa de ser una acció conjunta, si algun dels tres robots falla, l'objectiu no es pot assolir.

El primer robot és l'encarregat de guiar els altres dos, mitjançant un dispositiu GPS, i de realitzar el forat, mitjançant unes aspes. El segon segueix al primer, a través d'una càmera, i deixa la llavor al forat que ha fet el primer. I, per últim, el tercer robot segueix al segon, utilitzant dos sensors d'ultrasons, i tapa el forat fet pel primer. Tots els robots estan equipats amb sensors ultrasons i bumpers, per evitar xocar entre ells o altres objectes o animals. També estan equipats amb piles per alimentar els circuits, un pont en H i dos motors per l'adreçament dels robots.

Tota la programació del treball s'ha fet amb Arduino. S'han utilitzat quatre plaques, dos pel primer robot, una pel segon i una per l'últim robot.

Resumen

El principal objetivo de este proyecto es diseñar un sistema formado por tres robots que permitan reforestar terrenos de forma autónoma, dónde cada uno tendrá una función concreta pero no deja de ser una acción conjunta, si alguno de los tres robots falla, el objetivo no se puede llevar a cabo.

El primer robot es el encargado de guiar a los otros dos, mediante un dispositivo GPS, y de realizar el agujero mediante unas aspas. El segundo sigue al primero, a través de una cámara, y deja la semilla en el agujero hecho por el primero. Y, por último, el tercer robot sigue al segundo, utilizando dos sensores de ultrasonidos, y tapa el agujero hecho por el primero. Todos los robots están equipados con ultrasonidos y bumpers para evitar colisionar entre ellos u otros objetos o animales. También están equipados con pilas para alimentar los circuitos, un puente en H y dos motores para el direccionamiento de los robots.

Toda la programación del trabajo se ha hecho con Arduino. Se han utilizado cuatro placas, dos para el primer robot, una para el segundo y una para el último robot.

Abstract

The main objective of this project is to design a system formed by three robots that allow reforesting land autonomously, where each one will have a specific function but it is still a joint action, if any of the three robots fails, the objective cannot be carried out.

The first robot is responsible for guiding the other two, using a GPS device, and making the hole using blades. The second follows the first, through a chamber, and leaves the seed in the hole made by the first. And finally, the third robot follows the second, using two ultrasonic sensors, and covers the hole made by the first. All robots are equipped with ultrasound and bumpers to avoid colliding with each other or other objects or animals. They are also equipped with batteries to power the circuits, an H-bridge and two motors for addressing the robots.

All the code for the work has been done with Arduino. Four plates have been used, two for the first robot, one for the second and one for the last robot.

Agraïments

Després de 8 mesos de treball, avui per fi podem escriure els agraïments del treball de final de grau. Han sigut 8 mesos de treball dur, on el nostre dia a dia ha sigut realitzar les nostres 8 hores laborals i un cop acabades posar-nos amb el treball.

Volem agrair als nostres pares, mares, germans i parella per tot el suport que ens han donat en aquests mesos de desgast mental i físic, on sabem que sense ells hagués sigut molt més complicat el nostre dia a dia durant aquesta època.

També ens agradaria remarcar tot l'esforç que han realitzat els nostres pares per donar-nos tot el suport econòmic durant tot el nostre període universitari.

Per últim, volem aportar una menció al nostre tutor del treball, Joan Domingo, per la ràpida resposta cada vegada que necessitàvem un cop de mà i li desitgem una ràpida recuperació en el problema personal que va tenir en aquest últim mes. També volem agrair a Sebastián Tornil que hagi agafat el relleu del nostre director de treball i s'hagi implicat com si el treball l'hagéssim començat amb ell des d'un primer moment.

Glossari

Taula 1. Taula del glossari

Paraula	Descripció
Buffer	Memòria temporal per transmetre dades entre elements amb diferents velocitats
Rx/Tx	Pins de comunicació sèrie. Rx transmet dades. Per altre banda Tx fa de clock fent així el sistema síncron.
Clock	Senyal periòdica i binaria emprada per coordinar dispositius diferents
FPGA	De l'anglès Field Programmable Gate Array. És un dispositiu programable mitjançant llenguatges com VHDL.
GPL	De l'anglès General Public License. Llicencia gratuïta.
NMEA	De l'anglès National Marine Electronics Association. Es tracta d'un estàndard de comunicació establert entre les marques electrònica naval.
PCB	De l'anglès Printed Circuit Board. Es tracta de circuits electrònics implementats sobre una placa amb pistes de material conductor.
Protoboard	Placa per realitzar probes d'un circuit elèctric.
Trilateració	Tècnica de posicionament. Mitjançant un mínim de 3 balises i un senyal calcula la posició.
PWM	De l'anglès pulse-width modulation. Es tracta d'una tècnica per modificar el cicle de treball d'una senyal.
ICSP	De l'anglès In-Circuit Serial Programming.
RoHs	De l'anglès Restriction of Hazardous Substances. Es tracta d'una normativa que restringeix l'ús de certs materials en equips elèctrics i electrònics.

Índex

1. Prefaci

Des de fa molts anys, l'automàtica ha tractat d'abaratir els costos dels processos, eliminant les feines físicament més dures i les més perilloses. En aquest sentit, l'objectiu del nostre treball compleix ambdues, ja que pretén fer robots de baix cost que facin una feina de camp, que suposa un treball físic en condicions dures per l'ésser humà.

1.1. Origen del treball i motivació

L'origen del treball prové de la motivació per fer un treball que permetés realitzar feina de programació juntament amb el desig de fer quelcom que pogués suposar una ajuda mediambiental.

Amb aquestes dues idees clares, l'àmbit que més s'ajustava era l'automàtica i, entre els treballs proposats, vam trobar un del professor Joan Domingo que ens permetria programar robots per l'aplicació que proposarem.

Posteriorment i parlant sobre diferents temes va sorgir la idea de la reforestació com a tema. Permetia programar robots amb múltiples mòduls i útils, era mediambientalment útil i serviria per ajudar en un treball dur de camp.

1.2. Requeriments previs

Per a la realització d'aquest treball han estat necessaris els coneixements assolits al llarg de la carrera d'Enginyeria Electrònica, Industrial i Automàtica. Sent d'especial importància l'assignatura que introdueix l'automatització de processos, "Control Industrial i Automatització" (CIA), també és de clau importància l'experiència adquirida programant micros, treball realitzat durant "Informàtica Industrial" (IIEIA) i pel fet que la majoria del disseny es basa en electrònica digital, fent servir plaques FPGA (de l'anglès field-programmable gate array), també són de vital importància els coneixements adquirits a "Electrònica Digital i Microprocessadors" (EDMEIA).

La resta de coneixements s'han adquirit mitjançant la cerca d'informació. Com les plaques programables, l'entorn de sensors i actuadors de l'Arduino, llenguatge de programació de l'Arduino o tot el relacionat amb feines de reforestació.

2. Introducció

En aquesta introducció es parlarà dels objectius i de l'abast del treball.

2.1. Objectius del treball

L'objectiu d'aquest treball és la creació de tres robots de baix cost que treballin conjuntament per aconseguir automatitzar una part de les feines de reforestació. Es pretén realitzar un disseny hardware i software així com la fabricació d'un prototip funcional del sistema.

2.2. Abast del treball

Per a la realització del treball és necessari establir els requisits del sistema, així com una metodologia de treball encertada.

A continuació es numeraran els requisits del sistema:

- El sistema ha de ser econòmic, per la qual cosa es van utilitzar materials reciclats, sempre que fos possible, obtinguts del punt verd d'Esplugues de Llobregat. També s'han optat per plaques i sensors de baix cost.
- El sistema ha de ser capaç de funcionar en un entorn com el que es trobaria en una situació de reforestació real. Abans de realitzar la reforestació es neteja el terreny, apartant tot el que sigui possible d'apartar. Aquí el factor més crític serà la superfície de contacte, per això s'han observat diferents possibilitats comercials, escollint la que aparentment fos capaç de complir la funció amb el menor cost possible.
- Com realitzar un sistema capaç de reforestar qualsevol mena de planta és complexa i car, en aquest cas s'ha limitat el projecte als pins. Això es deu a que és dels pocs arbres que permet la seva sembra en situacions de reforestació amb una viabilitat acceptable i a l'abundància d'aquest arbre a la península Ibèrica.
- El sistema ha de ser capaç d'esquivar obstacles com els que es pot trobar, com pedres grosses difícils de retirar en treballs previs.
- El sistema ha de ser autònom, no obstant això, com s'escapa del nostre àmbit i coneixement, l'elecció dels punts on es plantaran els arbres queda a la responsabilitat de l'usuari.
- La precisió del sistema pot ser ample sempre que segueixi el patró establert per l'usuari i respecti unes mesures mínimes. Això es justifica en el fet que per a la viabilitat d'un bosc és important que els arbres guardin una distància mínima que depèn de l'espècie, però un error d'uns metres no suposa cap inviabilitat, ja que a mitjà i llarg termini, es pot observar per estudis

realitzats que el més important és que estiguin distribuïts pel terreny, encara que sigui de forma atzarosa.

Respecte a la metodologia de treball, es va començar plantejant el funcionament general del sistema. Un cop tenint clar això es van decidir les necessitats de cada robot i possibles solucions. Després es van adquirir els mòduls necessaris i vam procedir a la realització de proves amb les quals es van descartar aquells que van resultar ser més inestables. Amb la feina de proves feta, el disseny de hardware es podria considerar gairebé enllestit, restant per fer, el software, el muntatge dels estris i dels prototips.

El software es va testar tot mitjançant el monitor sèrie que proporciona la consola de l'Arduino, obtenint bons resultats. Però per raons de temps es va decidir començar el muntatge dels prototips abans de realitzar el disseny dels estris.

3. Estructura dels robots

Un cop està el terreny preparat per a la reforestació s'han de portar a cap tres accions. Primerament s'ha de fer un forat, tot seguit s'ha de col·locar una llavor en aquest forat i per últim s'ha de tapar el forat realitzat prèviament. Un cop sabent això s'ha de plantejar quants robots s'haurien de construir per poder realitzar aquesta acció i la decisió que s'ha pres és muntar un robot per cada acció, ja que instal·lar les tres funcions a un mateix robot dificultaria les seves reparacions, així com encariria el preu de la substitució en cas de ser necessari.

Per a la construcció dels robots, s'han reutilitzat cotxes teledirigits, va ser la solució més òptima que es va trobar. En relació a la part econòmica, ja que qualsevol cotxe pot ser reconvertit en un robot, i en relació a la facilitat, ja que el xassís està completament muntat, únicament s'ha de modificar l'electrònica. Tenint en compte el terreny en el qual treballarien els robots, s'ha escollit uns cotxes amb les rodes més adequades a aquesta situació. Però aquesta no va ser la primera idea.

El primer que es va plantejar a l'hora d'escollir com serien els robots van ser les rodes. La primera idea va ser utilitzar unes erugues, ja que són ideals per a terrenys muntanyosos, on no se sap com seran les condicions que poden haver-hi i perquè són fàcils de controlar. Consten dos motors, si s'ha de girar pares un i acciones l'altre. Aquesta opció es va descartar, ja que el cost era molt elevat per el projecte.

A part de les rodes, s'havia d'escollir el xassís del robot, es va plantejar la idea de fer el disseny 3D i després imprimir-ho, però aquesta idea també es va descartar pel preu que suposava la impressió de les peces i perquè el treball tindria un abast massa llarg per abordar-ho en el temps que es disposava.

Finalment, es va optar per l'opció presentada anteriorment, sabent que no era la més òptima però si la més viable.

3.1. Funcionament conjunt

Com ja s'ha comentat, l'objectiu d'aquests robots consisteix en realitzar una reforestació mitjançant tres robots.

El robot que va davant s'encarrega de fer el forat on anirà la llavor. Aquest està equipat per tres aspes, que s'ajunten a un punt on anirà connectat un motor que farà que giri per poder realitzar el forat de forma progressiva i no haver d'aplicar tanta força.

El segon robot està compost per un dipòsit on aniran totes les llavors, al final d'aquest dipòsit hi haurà una esfera amb dos forats amb la mida de la llavor, aquesta esfera girarà i deixarà caure una llavor i

quedarà col·locada de tal forma que a l'altre forat de l'esfera s'introduirà una altra, així amb mig gir es podrà realitzar la plantació.

Per últim, el darrer robot portarà un rastell, aquest té la finalitat d'arrosegar terra per poder tapar el forat fet prèviament.

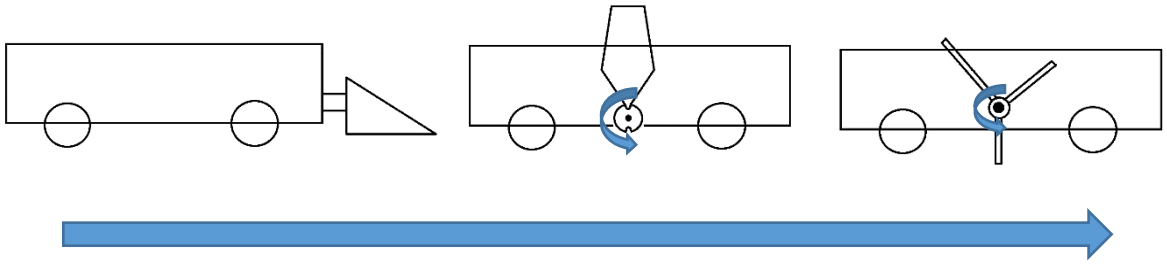

Figura 1. Representació gràfica del treball dels robots

4. Selecció de components electrònics

4.1. Dispositius per guiar el primer robot de forma autònoma

El primer dels tres robots, s'encarrega de guiar als altres, per tant, aquest s'ha de moure de manera autònoma. Per solucionar-ho s'han plantejat tres opcions.

La primera, col·locar un encoder a les rodes, per saber el nombre de voltes que realitza per anar d'un punt de plantació al següent. Degut que el treball es realitza a l'exterior i en un terreny muntanyós, no es pot preveure les condicions exactes, aquest sistema ha estat descartat, ja que les rodes podrien patinar i l'encoder contaria voltes falses.

L'altra opció que s'ha descartat per diverses raons, ha sigut la triangulació. La raó principal és que el treball es realitza a l'aire lliure, el sistema perd molta precisió. Un altra raó ha sigut l'elevat preu per realitzar aquesta instal·lació.

Finalment, l'opció escollida ha sigut el GPS, ja que la plantació de pins no necessita un elevat grau de precisió i és un dispositiu de baix cost. Dins d'aquesta opció, també va entrar la possibilitat d'utilitzar Galileo, que és el GPS europeu però finalment es va optar pel GPS que sembla un dispositiu més fiable.

4.2. Dispositius per detectar objectes

La idea principal per realitzar aquesta funció era fer una combinació de sensors. Aquest mètode consisteix en col·locar com a mínim tres sensors per mesurar la proximitat de l'objecte que hi ha davant i decidir parar el motor o reduir la velocitat d'aquest, depenent de les condicions, si com a mínim dos d'aquests sensors detecta, així s'assegura que realment hi ha un objecte davant i no és una mala lectura del sensor.

Finalment, es va decidir seguir una altra línia, es col·locaran tres sensors com s'ha pensat d'un inici, però dos de distància i un de contacte. El funcionament serà el següent, si el sensor de contacte detecta, els robots pararan automàticament, si un dels dos sensors de distància detecta es reduirà la velocitat si encara està a una certa distància i si els dos sensors de proximitat detecten, el robot esquivarà. Aquesta idea va sorgir per limitacions dels sensors de proximitat, ja que aquests si l'objecte està molt a prop, no mesuren de forma precisa.

4.3. Dispositius per el seguiment

Per aquesta funció (s'ha d'aplicar al segon i al tercer robot), s'han plantejat dues formes de fer-ho.

La primer opció es realitza a través d'una càmera, depenent de la posició que llegeixi la càmera, el robot s'haurà de moure cap a un lloc o cap a un altre. Aquesta opció és l'escollida pel segon robot, ja que es necessita més precisió que en el tercer, perquè aquest ha de deixar la llavor just on el primer ha fet el forat.

La segona opció, consisteix en utilitzar dos sensors de distància, si un dels dos deixar de detectar, es pot saber cap on s'està movent el robot de davant i així podrà decidir-se cap a on s'ha de moure el robot que porta equipats els dos sensors. Aquesta és la forma en la qual el tercer robot seguirà al segon, en aquest cas no es necessita tanta precisió, ja que únicament ha de tapar el forat fet pel primer empenyent sorra.

Encara que la càmera és l'opció més precisa, no s'ha decidit equipar els dos robots amb aquest dispositiu perquè és una opció elevada de preu i així es treballen dues formes diferents de fer un seguiment de robots.

4.4. Alimentació

Per alimentar als robots s'ha decidit fer-ho amb piles per davant de l'opció de les bateries, ja que amb les piles es pot arribar amb més exactitud al voltatge i amperatge desitjat. Amb la utilització de les piles també s'evita la instal·lació d'una bomba de càrrega.

4.5. Dispositius programables

A l'hora d'escollir la placa per a realitzar tota la programació, es va decidir utilitzar Arduino, ja que presenta una sèrie d'avantatges per davant de les altres plaques comercials.

Arduino té una gran comunitat. Genera una gran quantitat de documentació, gràcies al fet que aquesta plataforma és accessible per a tothom.

El seu entorn de programació és multi plataforma. Es pot instal·lar i executar en els sistemes operatius més comuns, Windows, MAC OS i Linux.

Utilitza un llenguatge de programació de fàcil comprensió. És un llenguatge basat en C++, que ajuda als nous programadors a iniciar-se però també permet una gran capacitat, on els programadors més avançats poden esprémer tot el seu potencial i adaptar-lo a qualsevol situació.

Aquestes plaques tenen un baix cost. Com que es tracta d'una plataforma oberta, les còpies de l'Arduino són molt habituals amb la qual cosa el preu dels diferents tipus de plaques són d'un cost molt

assequible per al consumidor. Encara que per a versions més definitives és recomanable comprar les versions oficials.

Re-usabilitat i versatilitat. Una vegada acabat el projecte és molt senzill de desmuntar les connexions externes i començar amb un nou projecte. La seva metodologia de connexió produeix un risc molt baix de fer una connexió errònia.

Per altre banda trobem plaques com Raspberry Pi. En aquest cas es tracten de mini ordinadors, son molt més potents que l'Arduino y no gaire mes cares. També compte amb una amplia comunitat que genera molta documentació. Per contra, aquestes plaques no s'acostumen a utilitzar en projectes d'aquest tipus. Fet pel que és més difícil trobar mòduls de sensors o actuadors dissenyats per a ella i consten de menys documentació. També és considerablement més gran que algunes de les plaques l'Arduino.

Tenint en compte això, a continuació es pot observar una taula resum.

Taula 2. Comparativa Raspberry vs Arduino

	Raspberry Pi	Arduino UNO R3	Arduino Nano
Mesura	56x65 mm	53x68 mm	45x18 mm
Preu/unitari (oficial)	38,82 €	24,20 €	24,20 €

5. Descripció de components electrònics utilitzats

Un dispositiu electrònic consisteix en la combinació de diversos elements organitzats en circuits, destinats a controlar i aprofitar les senyals elèctriques.

5.1. Arduino

Arduino és una plataforma de desenvolupament basada en una placa de hardware i software lliure. La placa incorpora un microcontrolador re-programable i una sèrie de pins femella, que permeten establir connexions entre el microcontrolador i els diferents sensors i actuadors d'una manera molt senzilla.

La placa Arduino és una PCB (Printed Circuit Board) que implementa un determinat disseny de circuit intern, d'aquesta forma l'usuari no ha de preocupar-se de les connexions elèctriques, directament pot començar a desenvolupar les diferents aplicacions electròniques.

5.1.1. Funcions bàsiques

A continuació, s'explicarà les dues funcions més bàsqiues de l'Arduino.

5.1.1.1. Setup

Aquesta és la primera funció que executarà l'Arduino i és obligatori escriure-la. La seva principal funció és iniciar els valors que utilitzarà el microcontrolador en cada programa.

5.1.1.2. Loop

Com en el cas del Setup, aquesta funció és obligatori escriure-la en el codi de programació. El comportament d'aquesta funció és el següent, un cop s'ha inicialitzat el programa, tot el que contingui aquesta funció s'executarà un nombre infinit de vegades, en resum és un bucle.

5.1.2. Arduino nano

En aquest cas s'ha optat per un Arduino nano ("Figura 2"), és una placa compacta, completa i compatible amb protoboards, basada en el microcontrolador Atmega328P. Té 14 pins d'entrada/sortida digital (dels quals 6 poden ser utilitzats amb PWM), 6 entrades analògiques, un cristall de 16 MHz, connexió Mini-USB, terminals per a connexió ICSP i un botó de reset.

Té les mateixes capacitats que una Arduino Uno però es veu retallat el seu connector USB, connector jack d'alimentació i els pins canvien a un format de pins header.

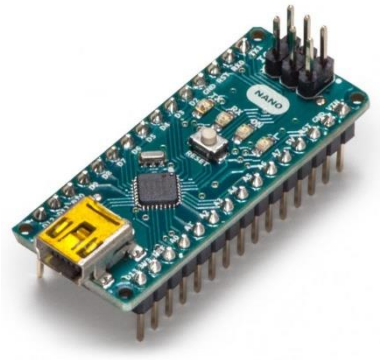

Figura 2. Arduino nano.

5.2. Sensor infraroig

Un dels sensors més utilitzats per a mesurar la distancia entre objectes és el sensor òptic. Per a fer el càlcul d'aquesta distància s'utilitza la triangulació ("Figura 3"), que consisteix en mesurar l'angle que forma un feix de llum entre l'emissor, l'objecte i el receptor.

L'emissor és una llum infraroja, que s'emet de forma puntual i a una freqüència determinada, fent que el receptor filtri i elimini qualsevol altra font de llum, com la llum del Sol o la llum ambiental. El receptor es un PSD, que detecta el punt d'incidència. Aquest punt depèn de la proximitat de l'objecte i de l'angle creat amb la triangulació.

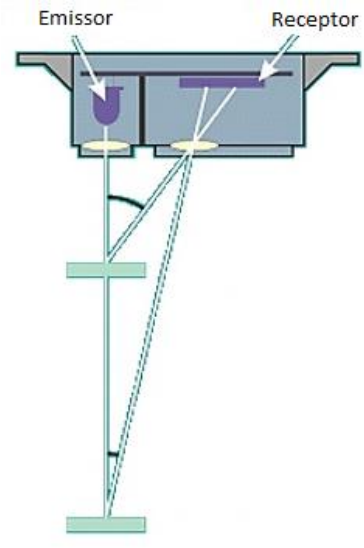

Figura 3. Funcionament del sensor infraroig.

5.2.1. SHARP GP2Y0A21YK0F

El rang de detecció d'aquest sensor és de 10 cm a 80 cm, fora d'aquest rang el sensor es comporta de manera inestable. Deixant de banda el rang estable del sensor, també s'ha de comentar que la sortida del sensor no és lineal ("Figura 4"), segueix una forma potencial negativa:

$$L = x \cdot z^{y} \qquad \text{[Eq. 1]}$$

On L és la longitud en cm, z és la lectura de l'ADC i la "x" i la "y" són les incògnites de l'equació.

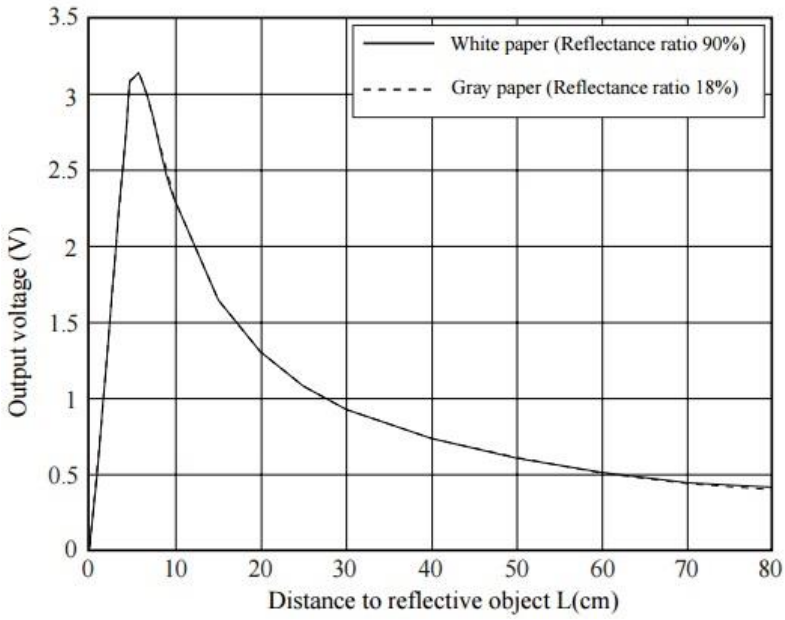

Figura 4. Sortida del sensor.

Perquè el sensor retorni les mesures en cm, és necessari saber l'equació de treball del sensor. Per aconseguir-ho es mesuren dos punts i es mesuren les distàncies a aquests punts amb un regle i es mira la lectura de l'ADC de l'Arduino, es realitza amb l'ADC per evitar fer un càlcul per poder obtenir el voltatge.

Objecte 1: La lectura de l'ADC de l'Arduino proporciona 269 quan la distància es de 20 cm.

Objecte 2: La lectura de l'ADC de l'Arduino proporciona 103 quan la distància es de 65 cm.

Realitzant un sistema d'equacions:

$$\begin{cases} 20 = x \cdot 269^{y} & \text{[Eq. 2]} \\ 65 = x \cdot 103^{y} & \text{[Eq. 3]} \end{cases}$$

S'aïlla una incògnita d'una equació:

$$x = \frac{65}{103^y} \qquad [\text{Eq. } 4]$$

Substituint a l'altra equació:

$$20 = 65 \cdot \frac{269^y}{103^y} \qquad [\text{Eq. } 5]$$

Resolen l'equació:

$$y = -1.2278 \qquad [\text{Eq. } 6]$$

I substituint a l'equació 4:

$$x = 19243.09 \qquad [\text{Eq. } 7]$$

Un cop realitzat el sistema d'equacions tenim que l'equació d'aquest sensor en particular segueix la següent funció:

$$L = 19243.09 \cdot z^{-1.2278} \qquad [\text{Eq. } 8]$$

Amb aquest sensor, encara que es deixi l'objecte a mesurar totalment quiet, els valors fluctuen com a conseqüència del soroll que es genera, per posar-li solució, es fa una mediana d'11 valors.

Després de moltes proves, aquest sensor ha estat descartat, per a les condicions d'ús d'aquest projecte és massa inestable i faria perdre precisió al robot.

5.2.2. Connexionat

El connexionat d'aquest circuit és molt senzill ("Figura 5"), s'ha de connectar la sortida de l'infraroig a una entrada analògica de la placa i per últim, connectar el V_{cc} als 5 V de la placa Arduino i fer comuns els GND dels dos dispositius.

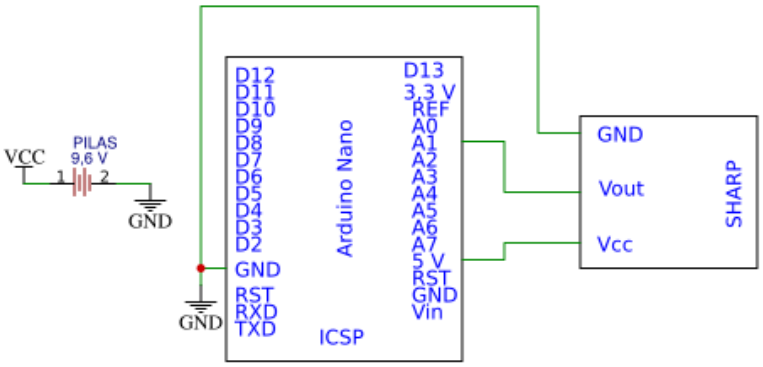

Figura 5. Connexionat del SHARP.

5.2.3. Diagrama de flux

En el diagrama ("Figura 5") s'explica el funcionament de l'infraroig, primerament es declara una variable a un valor de 0, tot seguit es llegeix el valor de l'infraroig mitjançant l'entrada analògica A1 de la placa, un cop llegit s'incrementa un valor la variable i. Aquest procediment es repetirà fins que el valor de la variable arribi a 11, quan això passi es farà una mitjana de tots els valors emmagatzemats.

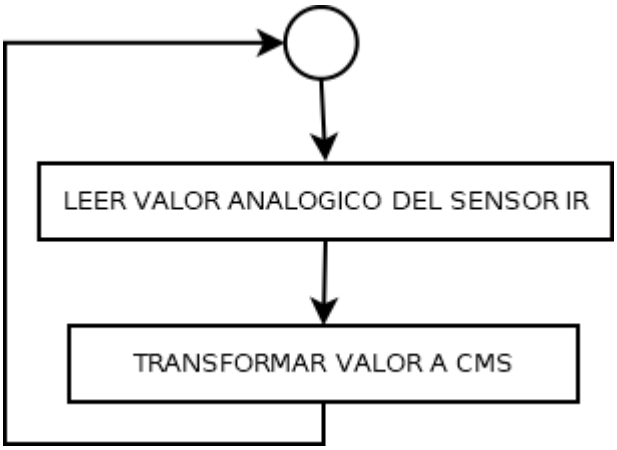

Figura 6. Exemple diagrama de flux del SHARP.

5.2.4. Codi del programa

```
#include <QuickMedianLib.h>

int IR[11];
int i;
int IR_med;

//****************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//****************************************************************//
```

```
void setup() {

}

//*****************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio bucle on s'executa la mesura de linfraroig amb una
mediana.
//*****************************************************************//

void loop() {
  i=0;
  while (i<11){
    IR[i]=19243,09 * (pow(analogRead(A1),-1.2278));     //Lectura del
valor analogic del sensor transformat a cms
    i=i+1;     //Augment de la variable i
  }
  IR_med=QuickMedian<int>::GetMedian(IR, 11);    //Amb els 11 valors
obtinguts es fa la mediana
}
```

5.3. GPS

La tecnologia GPS consisteix a aprofitar informació aportada per una constel·lació de 24 satèl·lits ("Figura 7") que estan a orbita. La informació rebuda és sobre la latitud, longitud, velocitat, temps... Aquesta informació rebuda per les estacions terrestres és unidereccional, únicament es rep i s'aprofita per realitzar càlculs però no es retorna cap resposta cap a la constel·lació. Aquests càlculs matemàtics són coneguts com a trilateració i en tot moment deu existir comunicació amb entre 4 i 12 satèl·lits.

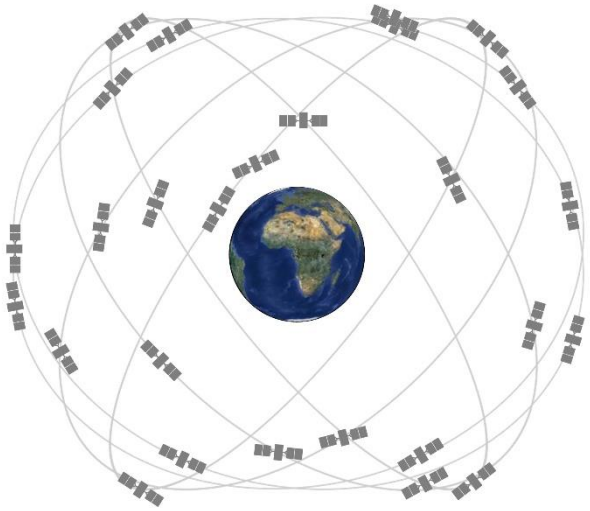

Figura 7. La terra envoltada dels 24 satèlits.

De tota la informació aportada per part dels satèl·lits del GPS, en aquest projecte s'aprofitarà la latitud i la longitud però a saber en tot moment on es troba localitzat el robot i cap a on ha d'anar. La unitat de mesura són els graus sexagesimals.

5.3.1. Latitud

Es defineix com la distància de qualsevol punt de la terra a una línia base horitzontal anomenada Equador. Aquests punts poden estar ubicats en dos grans zones, la zona que està per sobre de l'equador, la Nord i la zona que està per sota de l'equador, la Sud.

5.3.2. Longitud

Es defineix com la distància de qualsevol punt de la terra a una línia base vertical anomenada Meridià de Greenwich. Aquests punts poden estar ubicats en dos gran zones, la zona que està l'orient de la linia base, l'Est i la zona que està l'occident del meridià, l'Oest.

5.3.3. Recepció de dades

El GPS rebrà les dades en format NMEA, que es un protocol basat en linias de text. La linia comença amb un $, tot seguit del nom de la sentencia y després apareixen les dades (cadascuna separa per comes).

5.3.4. GPS-NEO6MV2

Aquest dispositiu ("Figura 8") és un mòdul GPS compatible amb Arduino. Té un petit format i un baix cost i consum. Té una precisió de 2.5 m parlant de la posició, de 0.1 m/s de velocitat i d'orientació, 0.5º. Amb aquestes especificacions, és considerat un dispositiu acceptable per a aquesta aplicació.

Figura 8. GPS-NEO6MV2.

5.3.5. Connexionat

El connexionat d'aquest circuit ("Figura 9") consisteix en alimentar la placa Arduino amb les piles i el GPS amb la sortida de 5 V de l'Arduino. Un cop alimentades i amb les masses comunes, es connecten els pins Rx i Tx a dos pins digitals, amb l'objectiu de realitzat la comunicació.

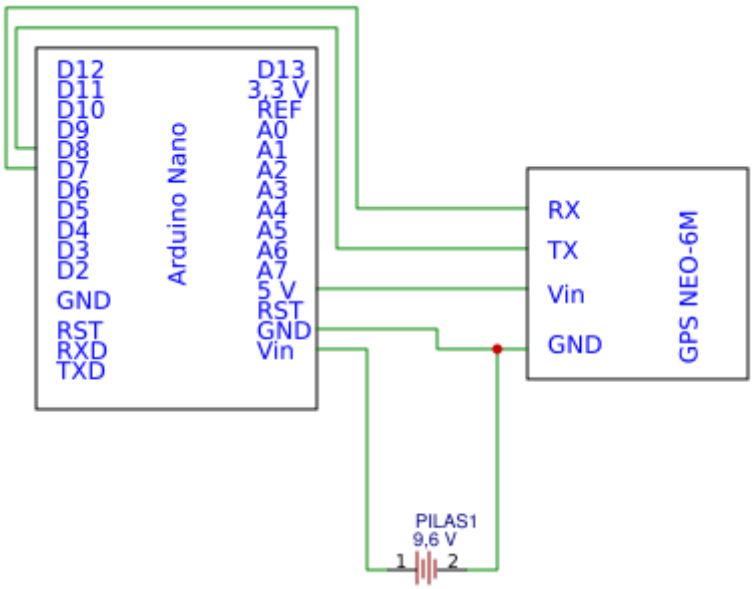

Figura 9. Connexionat pont en H.

5.3.6. Diagrama de flux

En el següent diagrama ("Figura 10") s'explica el funcionament del codi del GPS. On s'estableix la velocitat de transmissió, es llegeix el codi NMEA i s'emmagatzema el valor de la latitud.

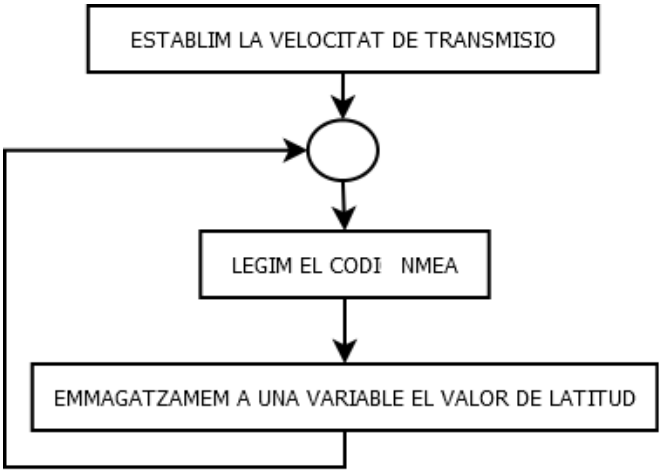

Figura 10. Diagrama de flux, codi GPS

5.3.7. Codi del programa

```
//Llibrerias
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>


SoftwareSerial mySerial(8, 7); //Pins de comunicació amb el GPS
Adafruit_GPS GPS(&mySerial);

double latitud_actual; //Varible per emmagatzemar

void setup() {
  //Estableix velocitat de transmissió
  GPS.begin(9600);
  // Activa el RMC (informacio minima recomenada) i el GGA (correcio de
dades si no se sap el valor de altitud)
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  // La frequencia a la que se actualitza el dispositiu es de 1 Hz, no es
recomana usar mes de 1 Hz, ya que podria ser que no dones tiempo a
ordenar dades i mostrarlos
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);

}

void loop() {

  char c = GPS.read(); //Llegeix el codi NMEA
  latitud_actual=GPS.latitude;  //Emmagatzema la variable latitude

}
```

5.4. Pont en H

Aquest circuit està format per 4 transistors (“Figura 11”) connectats a V_{cc} i GND. La càrrega (dispositiu a controlar), es troba entre els transistors. Els transistors actuen com interruptors, gràcies a això es pot variar el sentit de la corrent que circula per la càrrega. Una de les coses a tindre en compte és que no es poden activar els dos transistors d’una mateixa branca, ja que es provocaria un curtcircuit entre GND i V_{cc}.

En el cas d’aquest projecte, la càrrega és un motor. En un motor corrent continua, quan canvies el sentit del corrent que circula per aquest, fa que la polaritat del motor també canviï i així girar en el sentit oposat al que ja girava.

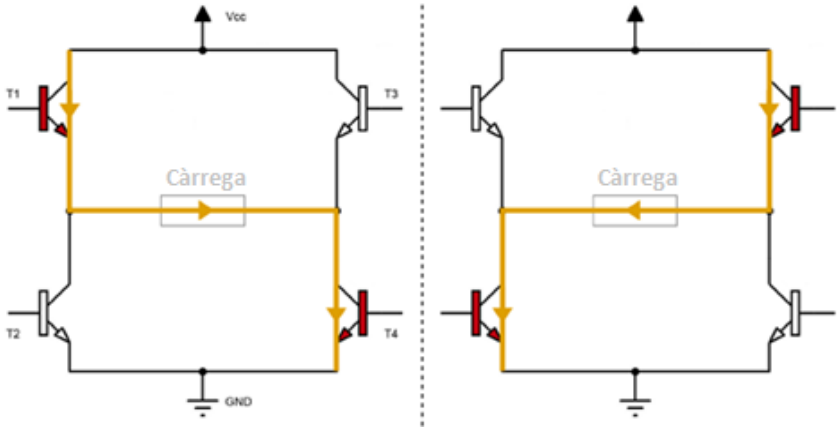

Figura 11. Circuit intern d'un pont en H.

5.4.1. LN298N

El LN298N ("Figura 12") és un dispositiu que integra dos ponts en H, això fa que es pugui controlar el sentit de gir de dos motors amb un sol mòdul. A part de controlar el sentit de gir, amb aquest circuit integrat també es pot controlar la velocitat de gir de cadascun dels motors introduint un senyal PWM.

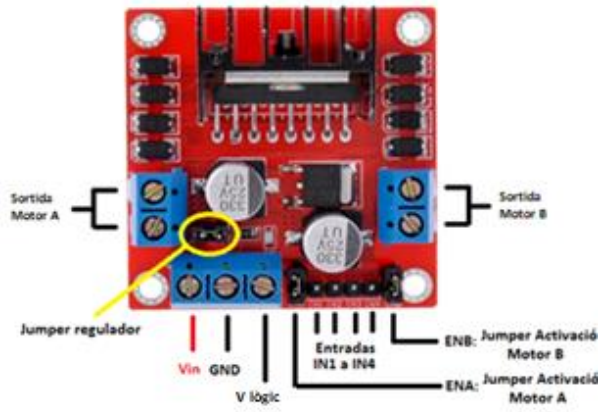

Figura 12. LN298N.

Per a fer un ús correcte del LN298N s'ha de tindre uns aspectes en compte:

- Els motors rebran 3 V menys que la tensió amb la qual alimentem el mòdul, ja que l'electrònica d'aquest els consumeix.
- Per activar el regulador de tensió, és necessari tancar el jumper del regulador, d'aquesta forma tindrem una sortida lògica de 5 V, que es pot utilitzar amb total llibertat (es pot fer servir per alimentar la placa Arduino).
- Amb el regulador alimentat amb un jumper, aquest només funciona fins a 12 V, si s'ha d'aplicar més tensió és necessari alimentar la part lògica amb un altra font externa.
- Si es treu el jumper és necessari alimentar la tensió lògica amb 5 V.

- Si el jumper del regulador està activat i s'alimenta la tensió lògica, es podria fer malbé el mòdul.
- El senyal PWM, per regular la velocitat de gir dels motors, es por introduir en els pins ENA si se'ls hi treu els jumpers d'activació o si aquest estan col·locats, es pot proporcionar el PWM en un dels dos INPUTS del motor i l'altre INPUT en estat baix. En el cas d'aquest projecte s'ha decidit la segona opció, ja que així el connexionat se simplifica.

5.4.2. Connexionat

A la següent figura es mostra el connexionat del LN298N ("Figura 13"), amb dos motors i l'Arduino nano. En el cas d'aquest projecte, s'ha decidit no utilitzar els pins ENA del dispositiu del control de la velocitat del motor, el que es fa és modificar el PWM d'una de les dues entrades procedents de l'Arduino. Els inputs 1 i 2, pertanyen al motor A i estan connectats als pins 9 i 10 de l'Arduino respectivament, i els inputs 3 i 4, pertanyen al motor B i estan connectats als pins 11 i 12 de la nano.

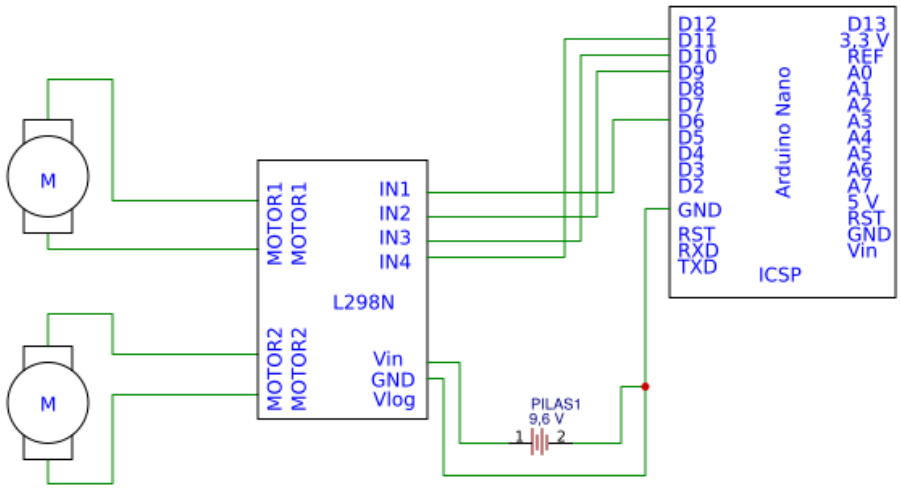

Figura 13. Connexionat del LN298N.

5.4.3. Diagrama de flux

En el següent diagrama ("Figura 14") s'explica el funcionament del pont en H mitjançant un exemple d'ús senzill.

El moviment dels motors està controlat a través de 2 pins cadascun, un en estat alt i l'altre en estat baix. A l'exemple que es veu a continuació reflecteix el cas on el robot anirà cap endavant i a la dreta, esperarà 5 segons en aquest sentit i canviarà a l'estat oposat, endarrere i a l'esquerre.

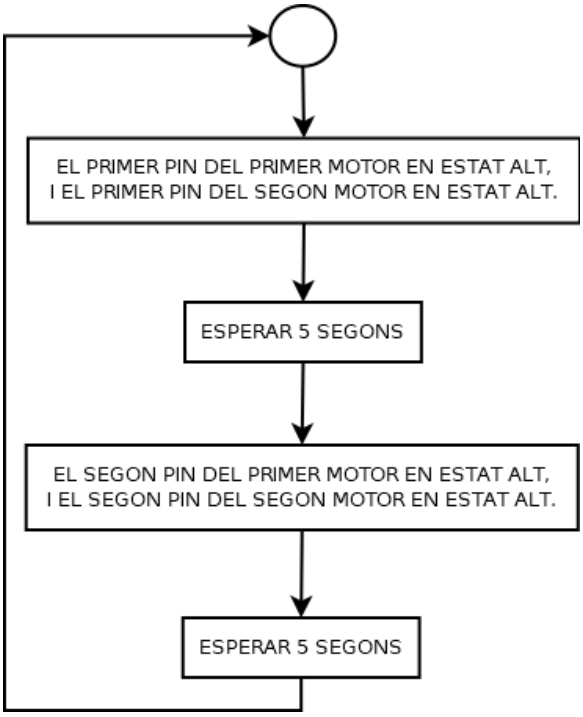

Figura 14. Exemple del diagrama de flux del LN298N

Si es vol que aquesta velocitat sigui variable o no sigui el màxim que pot donar el motor, es posa un pin en estat baix i l'altre s'assigna a un valor de PWM o a una variable.

5.4.4. Codi del programa

```
#define pinIN1A 6   //Pin que acciona motor de darrere
#define pinIN2A 9   //Pin que acciona motor de darrere
#define pinIN1B 10    //Pin que acciona motor de davant
#define pinIN2B 11    //Pin que acciona motor de davant

//**********************************************************************
**********************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//**********************************************************************
**********************************************************************//

void setup() {

  pinMode(pinIN1A, OUTPUT);
  pinMode(pinIN2A, OUTPUT);
  pinMode(pinIN1B, OUTPUT);
  pinMode(pinIN2B, OUTPUT);

}

//**********************************************************************
**********************************************************************//
// Nom de la funcio: loop()
```

```
// Funcio: Funcio encarregada d'activar els motors per direccionar el
robot amb velocitat variable.
//*********************************************************************
*********************************************************************//

void loop() {

  //Primer pin del primer motor en estat alt, primer pìn del segon motor
en estat alt
  digitalWrite(pinIN1A, HIGH);
  digitalWrite(pinIN2A, LOW);
  digitalWrite(pinIN1B, HIGH);
  digitalWrite(pinIN2B, LOW);
  delay(5000);

  //Segon pin del primer motor en estat alt, segon pìn del segon motor en
estat alt
  digitalWrite(pinIN1A, LOW);
  digitalWrite(pinIN2A, HIGH);
  digitalWrite(pinIN1B, LOW);
  digitalWrite(pinIN2B, HIGH);
  delay(5000);
}
```

5.5. Sensor d'ultrasons

Aquests sensors fan servir ones sonores d'alta freqüència ("Figura 15"), a partir de 20 kHz, no audibles pels éssers humans amb l'objectiu de detectar obstacles.

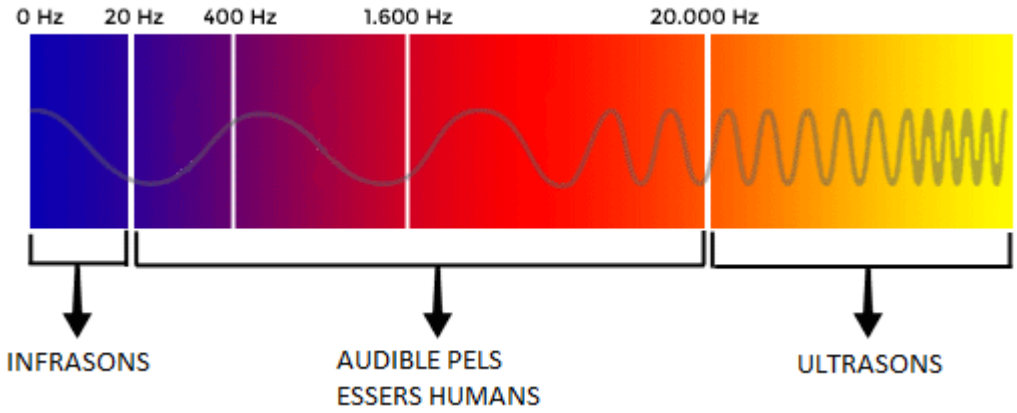

Figura 15. Tipus de ones en relació a la freqüència emprada.

El mètode de funcionament ("Figura 16") consisteix a calcular el temps que triga l'ona, enviada pel sensor, en anar i tornar després de rebotar contra un objecte.

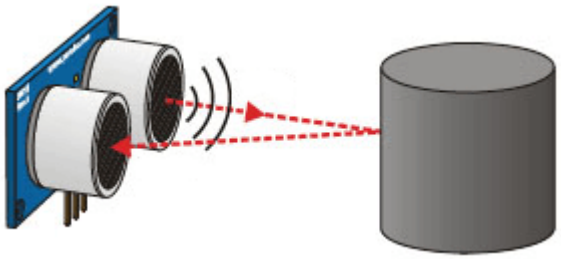

Figura 16. Funcionament ultrasò.

Així doncs, per calcular la distància només s'ha d'aplicar la formula

$$L = \frac{t \cdot c}{2}$$

On L és la distancia en cm, t el temps en s i c la velocitat del so en cm/us.

Aquest sensor és especialment útil, ja que no es veu afectat per la transparència de l'objecte, per la forma de l'objecte, ni per la pols o la brutícia. Aquest últim punt, és especialment útil per realitzar el treball, ja que els robots s'hauran de moure per entorns bruts i polsegosos típics de les zones a reforestar.

5.5.1. HC-SR04

Hem optat pel sensor HC-SR04 ("Figura 17") a causa del fet que, tot hi el seu reduït preu, compleix les necessitats. Teòricament mesura de 2 a 400 cm. A la realitat ens hem trobat que presenta lectures inestables a partir de 200 cm, la qual cosa no és un problema, ja que a la velocitat dels robots no és necessari reaccionar a objectes que trobem a aquestes distàncies.

Figura 17. HC-SR04.

5.5.2. Connexionat

A la figura següent es mostra el connexionat del HC-SR04 amb la placa Arduino ("Figura 18"). S'ha decidit fer servir l'alimentació de sortida de la placa Arduino (pin 5 V) que el controla tenint en compte que el sensor consumeix 15 mA quan està en funcionament i l'alimentació del pin 5 V de l'Arduino pot suportar fins a 200 mA.

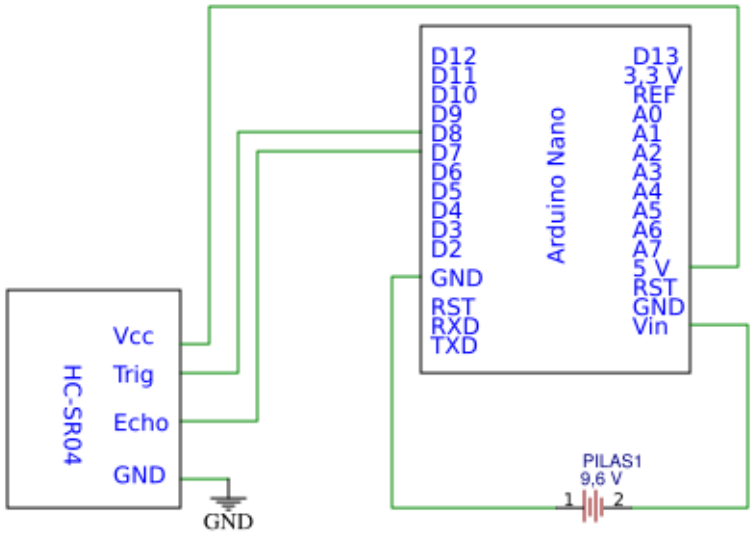

Figura 18. Conexionat de l'ultrasons.

5.5.3. Diagrama de flux

Al següent diagrama ("Figura 19") es mostra el funcionament del sensor HC-SR04.

Primer es comprova no haver emès cap senyal, ja que crearia interferències amb el senyal que controlem. Per això es deixa el pin TRIG a 0 durant un us. Un cop passats aquest petit temps s'envia un senyal d'ultrasons pel pin TRIG, s'espera 3 us i es torna a deixar el pin TRIG a 0. Seguidament es llegeix el sensor pel pin ECHO.

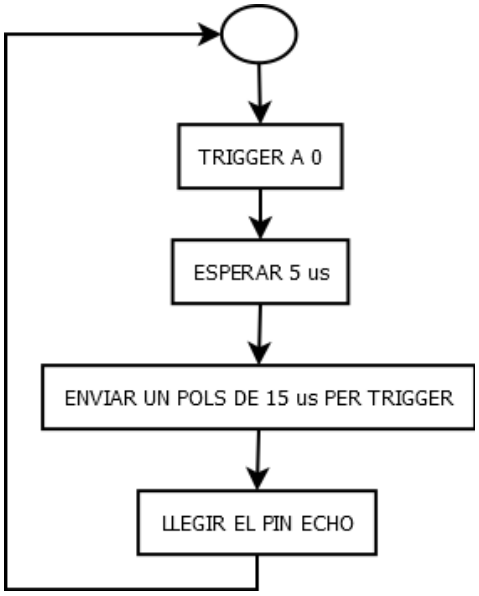

Figura 19. Exemple del diagrama de fluxe del HC-SR04.

5.5.4. Codi del programa

```
//Llibrerias
#include "QuickMedianLib.h"//Para tratamiento de datos

//Ultraso
#define ECHOPIN 7 // Pin echo, encarregat de rebre la señal
#define TRIGPIN 8 // Pin trig, encarregat de enviar la señal

//Variables ultraso
int distancia_sonido[11];
int med_ultra;
int distancia;

//*****************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//*****************************************************************//

void setup(){
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
}

//*****************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio bucle on s'executa la mesura de lultraso amb una
mediana.
//*****************************************************************//

void loop() {

  int i = 0;
  while (i <= 11) {
```

```
    digitalWrite(TRIGPIN, LOW);   //Pin de dispar a esta baix
    delayMicroseconds(10);
    digitalWrite(TRIGPIN, HIGH);    //Envia un estat alt per tal de saber
on es l'objecte
    delayMicroseconds(30);
    digitalWrite(TRIGPIN, LOW);   //Es torna a posar el valor del trigger
a 0
    int distancia = pulseIn(ECHOPIN, HIGH);   //Llegueix el temps que ha
tardat en retornar el estat alt enviat anteriorment
    distancia = distancia / 58;   //Calcula la distancia del pols
(anteriorment es mesurava en temps)
    if (distancia>0){
      distancia_sonido[i] = distancia;    //S'emmagatzema 11 valors de
distancia
      i = i + 1;
    }
  }
  med_ultra = QuickMedian<int>::GetMedian(distancia_sonido,
11);    //Mitjancant la libreria es treu la mediana d'aquests 11 valors
mesurats anteriorment
}
```

5.6. Sensor d'imatge

La millor opció disponible per aconseguir precisió seguint a Iron és la visió artificial. Arduino disposa de moltes càmeres i de diverses llibreries per fer-les servir. No obstant això, no totes les càmeres són bones opcions. Existeixen càmeres sense buffer que són molt complexes d'aconseguir fer anar amb Arduino, per la velocitat necessària per emmagatzemar els píxels. Existeixen opcions comercials que si consten d'aquest buffer, entre aquestes opcions es troba la PixyCam ("Figura 20").

5.6.1. PixyCam 2.0

Es tracta d'una càmera de software i hardware lliure amb llicència GPL, integrada a una PCB. Està preparada per comunicar-se amb Arduino o altres plaques similars des del moment que es treu de la capsa connectant-se a través dels pins ICSP.

A aquesta càmera és possible ensenyar-li a identificar un objecte mitjançant un software. Les principals característiques emprades per aquest fi són la forma, i el color. Raó per la qual s'ha decidit fer servir una pilota de tenis com objecte a seguir. La Pixy és capaç de detectar un objecte de 4x1 píxels segons les seves especificacions. No obstant això hem pogut observar que si allunyem la pilota a un metre aproximadament, les lectures comencen a ser més irregulars pel que fa detectar la forma, i té lectures falses.

Respecte a la detecció, té un rang de 60º de visió en l'eix horitzontal i 40º en l'eix vertical.

Un cop emmagatzemat a la memòria de la càmera, és possible accedir a les dades del bloc mitjançant llibreries. D'aquesta manera podem saber la posició a la que es troba l'objecte respecte a l'extrem esquerre de la imatge.

Figura 20. PixyCam 2.0.

5.6.2. PixyMon v2

Es tracta d'un software per ensenyar a la càmera a reconèixer objectes. Tot hi que aquest procés es pot dur a terme sense el software, només pitjant el polsador de la càmera, es molt recomanable utilitzar-lo.

Connectant la càmera a un ordinador mitjançant un USB i obrint el software començarem a veure el que la càmera veu ("Figura 21").

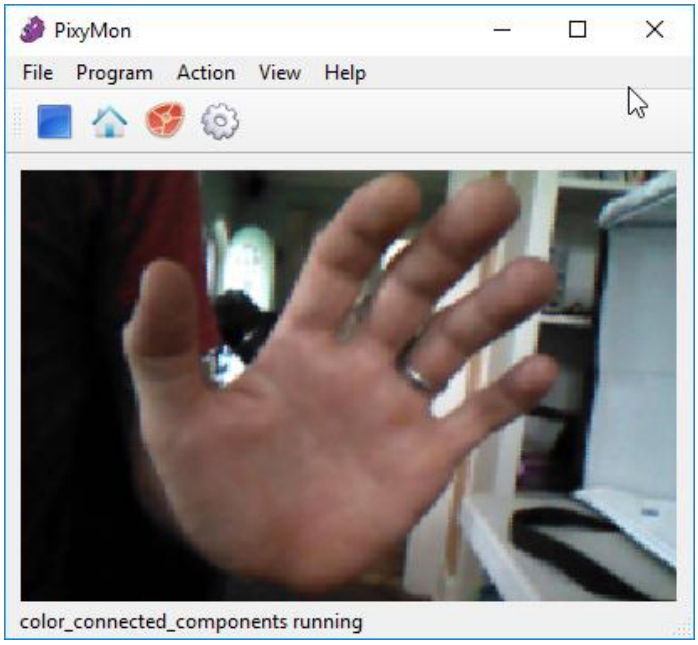

Figura 21. Exemple del software PixyMon v2 quan es connecta la càmera Imatge obtinguda de la pagina oficial de Pixy.

Seguidament fent clic a "Action->Set signature 1..." ("Figura 22"), podrem ensenyar a la càmera un objecte, que quedarà enregistrar a la seva primer posició.

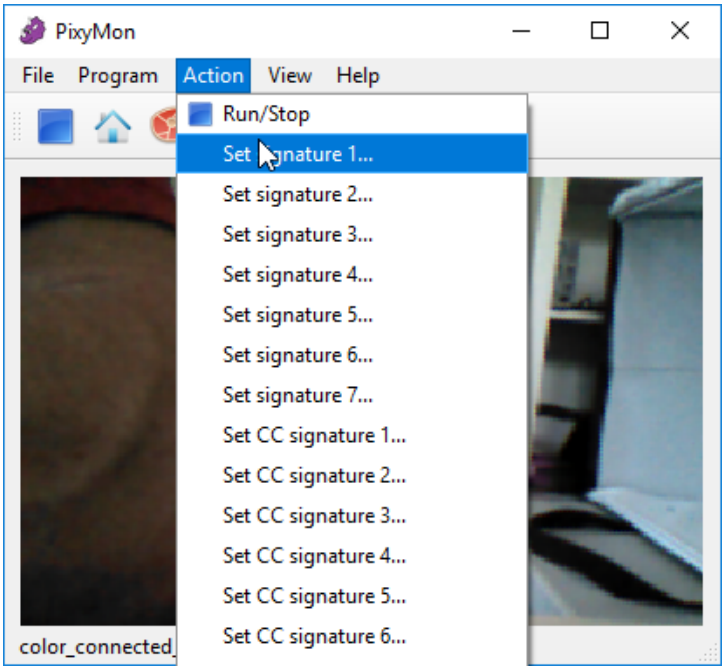

Figura 22. Imatge il·lustrativa de com seleccionar la posició de memòria on guardar l'objecte. Imatge obtinguda de la pagina oficial de Pixy.

Fent servir el ratolí, com es mostra a la ("Figura 23"), es selecciona l'àrea que la càmera farà servir per reconèixer l'objecte.

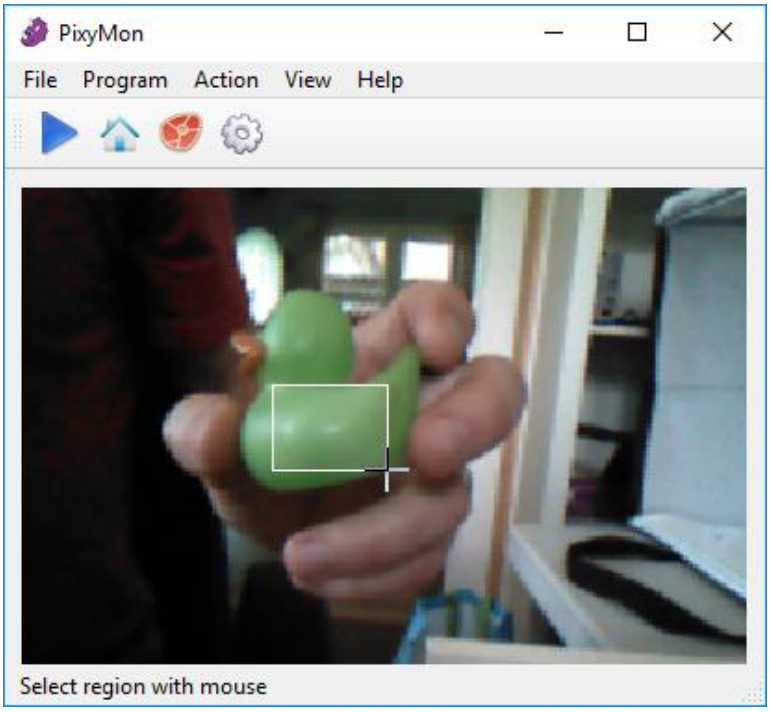

Figura 23. Selecció d'àrea per ensenyar a reconèixer l'objecte. Imatge obtinguda de la pagina oficial de Pixy.

En la següent imatge ("Figura 24") es veu com l'objecte queda registrat.

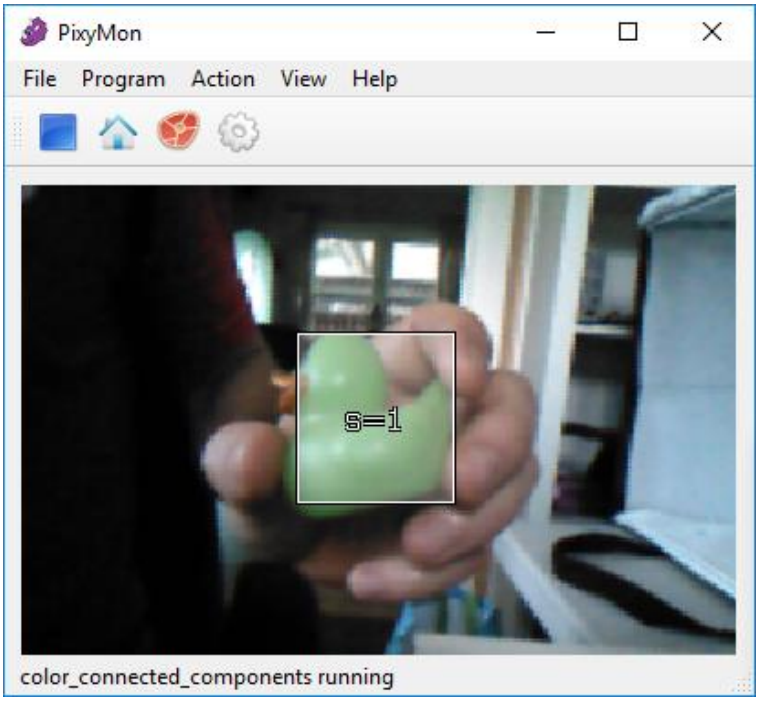

Figura 24. Exemple d'objecte registrat. Imatge obtinguda de la pagina oficial de Pixy.

Es pot intentar millorar la detecció de l'objecte, en quant a quantitat de deteccions falses amb l'opció de configure que aporta el PixyMon ("Figura 25").

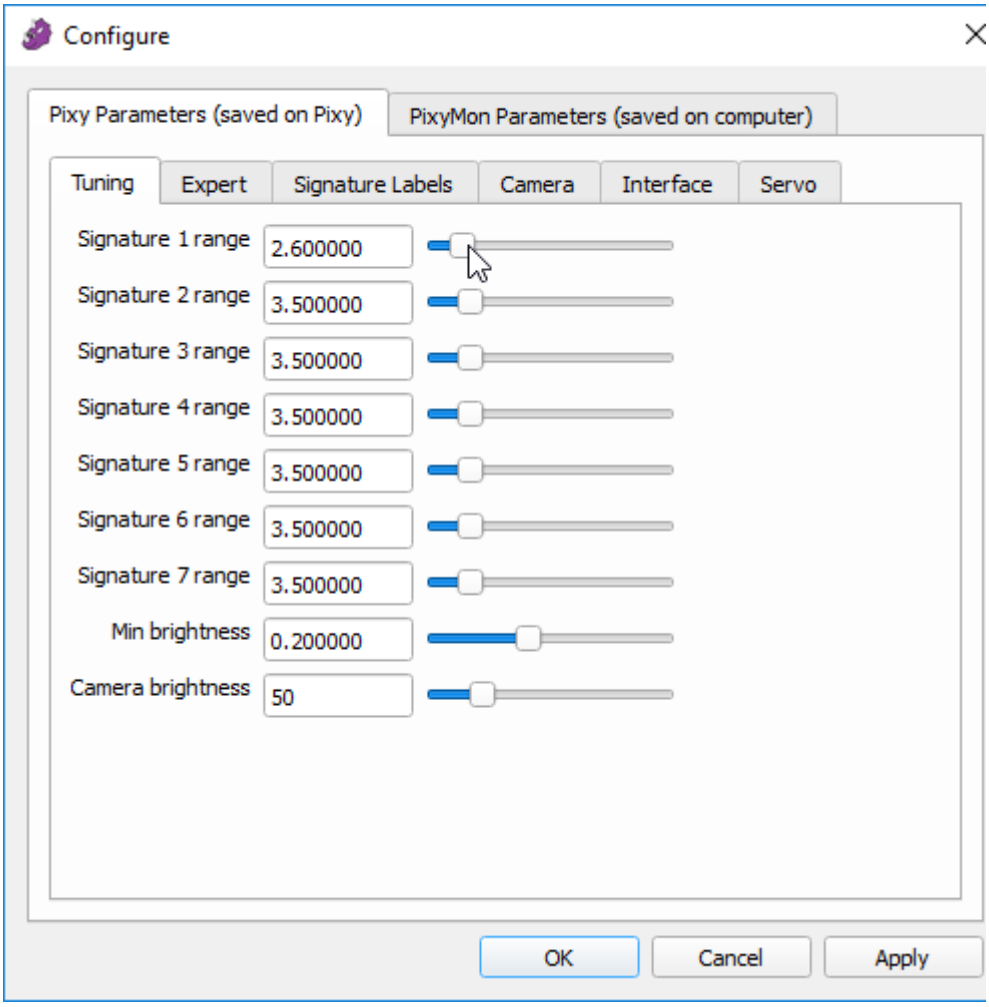

Figura 25. Paràmetres de configuració per millorar la precisió de detecció de la càmera. Imatge obtinguda de la pagina oficial de Pixy.

5.6.3. Connexionat

El connexionat de la càmera (“Figura 26”) es portarà a cap mitjançant els pins ICSP, és una connexió directa.

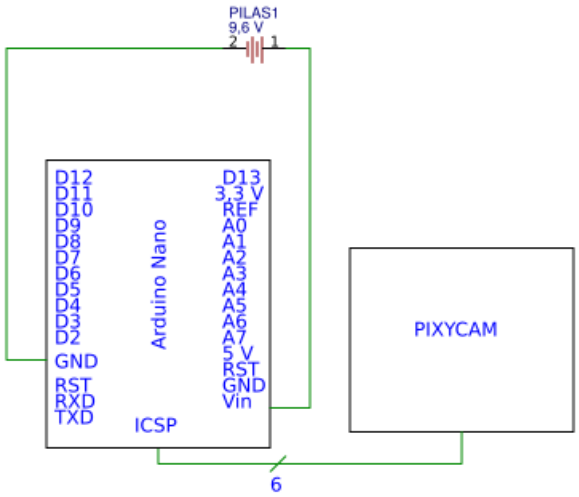

Figura 26. Conexionat càmara.

5.6.4. Diagrama de flux

A continuació es mostra el diagrama de flux ("Figura 27") emprat per al funcionament de la càmera PixyCam2 al robot Groot.

Primerament s'activa la càmera, s'habilita les característiques de l'objecte, després comprova aquests blocs detectats i retorna a l'Arduino la característica d'interès, en aquest cas la posició X.

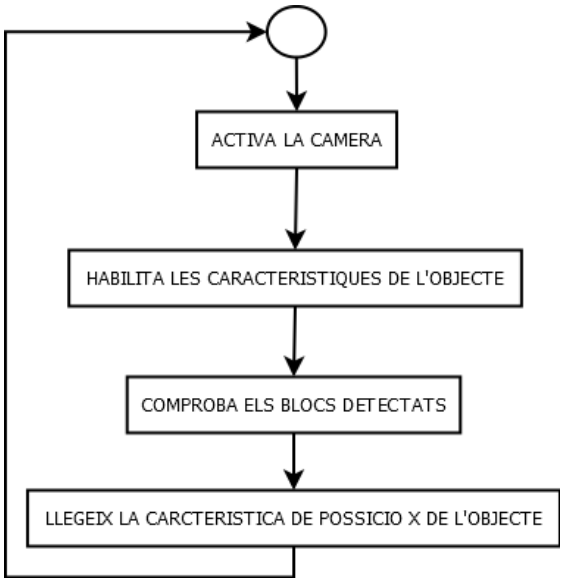

Figura 27. Exemple del diagrama de flux PixyCam2.

5.6.5. Codi del programa

```
#include <Pixy2.h>    //Camara
Pixy2 pixy;
```

```
//*********************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//*********************************************************************//

void setup()
{

  Serial.begin(115200);

  //Inicialitza la camara
  pixy.init();
}

//*********************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio encarregada de la detecció de la posició X d'un objecte
mitjançant la camera
//*********************************************************************//

void loop()
{
  int posX;

  pixy.ccc.getBlocks();

  if (pixy.ccc.numBlocks)   //Si detecta la camara...
  {
    //S'extreu el valor de la posicio x
    posX=pixy.ccc.blocks[0].m_x;
  }
}
```

5.7. Piles

Per la realització d'aquests robots s'ha optat per fer servir piles recarregables per alimentar els robots. Això permet controlar de forma molt senzilla tant el voltatge com el pes, factors importants, ja que:

- Algun dels motors requereix força per poder fer un forat.
- El pes ajudarà a evitar que el robot s'aixequi quant faci el forat
- També pel seu preu, invertint 1 € en bateries es poden aconseguir 800 mA, amb les piles comprades equival a 1.362 mA.
- També s'ha valorat l'aspecte mediambiental, per aquest motiu s'han escollit piles recarregables, de llarga durada i una eficiència elevada.

5.7.1. Amazon basics

Aquestes piles (“Figura 28”) han estat les majoritàries a l’hora d’alimentar els robots. Són còpies d’unes piles d’alta eficiència, també emprades en aquest treball.

Figura 28. Piles de amazon basics.

5.7.2. Connexionat

Connectant 8 piles d’1,2 V (“Figura 29”), s’obté una font d’alimentació de 9,6 V, amb la que s’alimentarà tot el conjunt de cada robot. La connexió és la que es pot apreciar a la següent imatge, és un connexionat en sèrie, fent que el voltatge de cada pila se sumin.

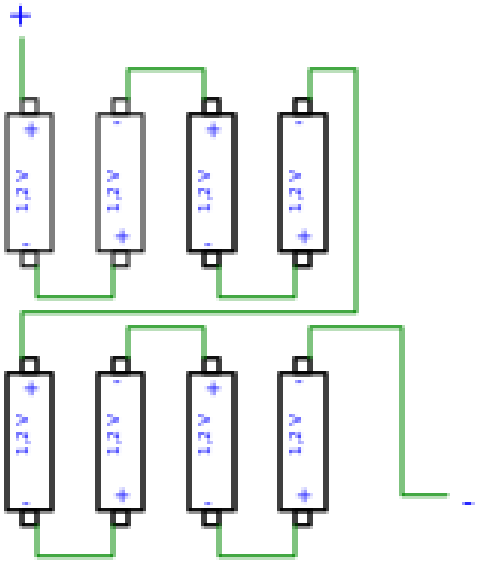

Figura 29. Connexionat piles.

5.8. Bumper

Aquest dispositiu ("Figura 30") consisteix bàsicament en un switch, si s'activa detectarà massa i el robot s'aturarà indefinidament, ja que en cas de col·lisió, significaria que els altres sensors no han estat capaços de detectar l'obstacle, es podrien haver fet malbé i seria necessari canviar-los. És una mesura preventiva.

Figura 30. Bumper de contacte.

5.8.1. Connexionat

En aquest esquema (“Figura 31”) apareix l’alimentació de la placa Arduino i el connexionat del bumper. Alimentat amb 5 V i un divisor de tensió format amb una resistència de 1 kΩ.

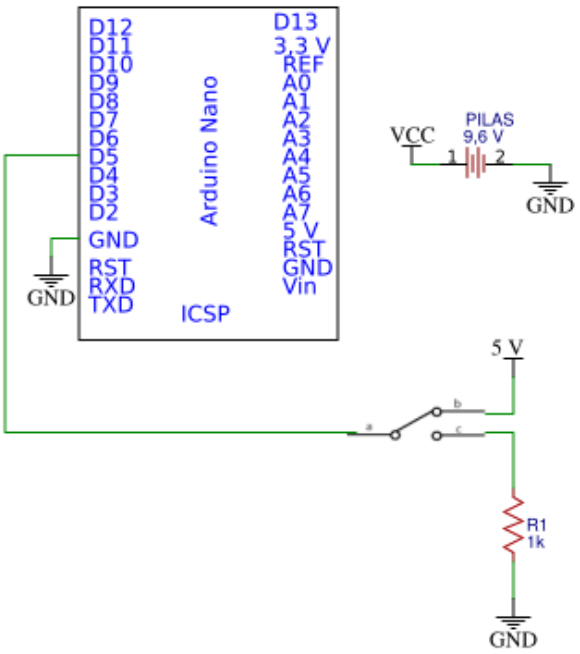

Figura 31. Connexionat bumper.

5.8.2. Diagrama de flux

Aquest sensor es fa servir com interrupció. A continuació es mostra el diagrama de flux (“Figura 32”) emprat com a interrupció. En el moment que l’Arduino rebi un estat HIGH, voldrà dir que el bumper a rebut un contacte i el PWM de tots els motos passa a ser 0, per parar completament el robot.

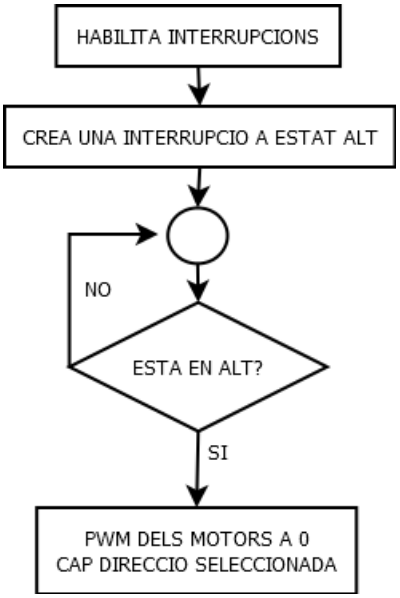

Figura 32. Exemple del diagrama de flux Bumper.

5.8.3. Codi del programa

```
#include <TimerOne.h>   //Timer

//*****************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//*****************************************************************//

void setup() {
  //Inicializacion Timer
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(bumper);
  attachInterrupt(digitalPinToInterrupt(2),bumper,RISING);
}

//*****************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio d'interrupcio per aturar el robot si colisiona contra
algun objecte.
//*****************************************************************//

void loop(){
  salida=0;
  speed=0;
 }
```

6. Disseny i programació dels robtos

L'objectiu de reforestar un terreny es porta a cap amb tres robots, cadascun amb una funció determinada. A part de les funcions pràctiques que realitzen els robots també realitzen de manera diferent la decisió de quan i cap a on s'han de moure, això significa, estan equipats de manera diferent, parlant tant d'útils com de sensors.

6.1. Iron

L'Iron ("Figura 33") és l'encarregat de fer la primera acció de la reforestació, la seva funció és la de realitzar un forat on anirà la llavor i a part d'això serveix de guia per als altres dos robots.

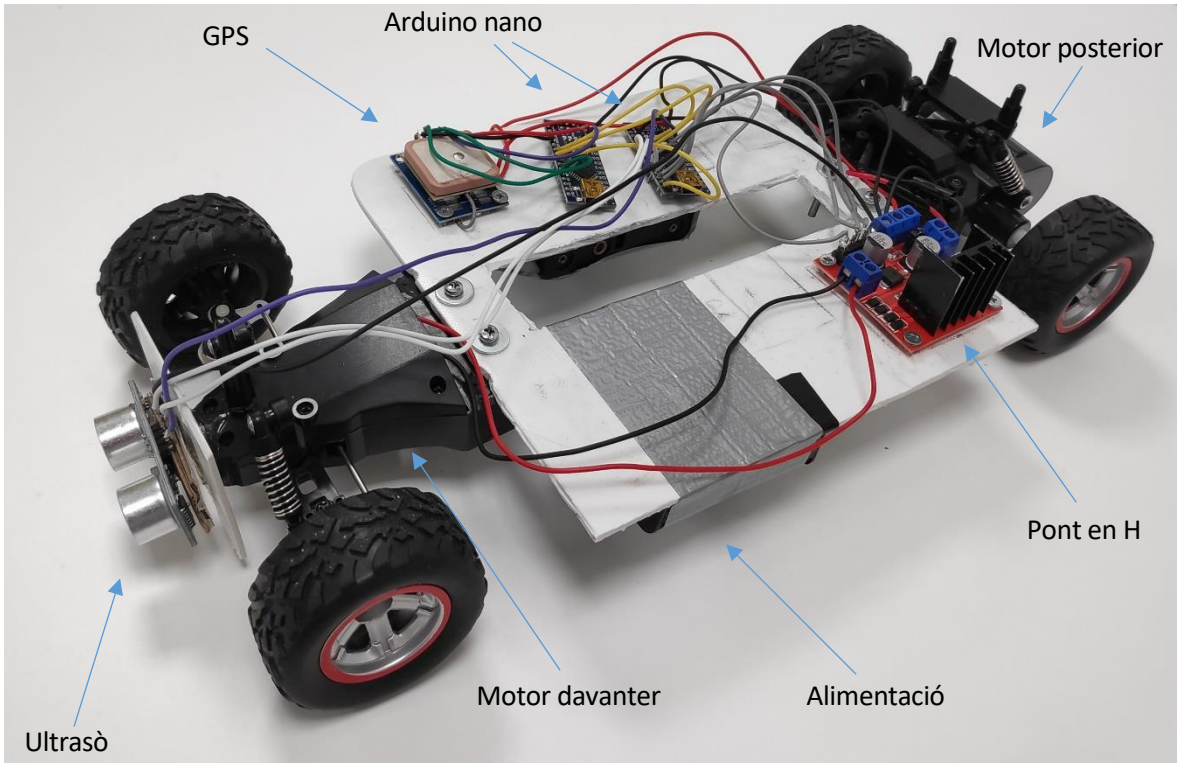

Figura 33. Imatge del primer robot.

6.1.1. Mecànica

L'útil d'aquest robot consisteix en tres aspes enganxades a un motor que les fa girar. Amb aquest gir el que es pretén és que amb menys esforç es pugui fer el forat, ja que no es realitza el forat de cop, sinó que es fa per capes amb les passades de les aspes que cadascuna és més llarga que l'anterior.

6.1.2. Electrònica

Electrònicament, aquest és el robot més important dels tres, ja que si aquest deixés de funcionar, els altres també ho farien.

Està format per un dispositiu GPS, encarregat de donar l'ordre de cap a on s'ha de moure en cada moment, un pont en h, que és l'encarregat de fer girar els motors del cotxe en diferents direccions per tal de poder tindre a l'abast tots els moviments necessaris, i per últim dos sensors de seguretat per tal de no xocar amb cap objecte o animal a l'hora de reforestar, aquests sensors són un d'ultrasons i un bumper, aquest últim és l'encarregat de parar el cotxe si actua, ja que significaria que alguna cosa no està funcionant correctament perquè els altres sensors no han detectat.

6.1.3. Connexionat

En la següent imatge ("Figura 34") es pot apreciar l'esquemàtic del primer dels tres robots. Les dues plaques Arduino, alimentades cadascuna amb 9.8 V, igual que el pont en H. Aquest robot també consta del GPS i un ultrasò, alimentats amb 5 V extrets de l'Arduino. Per últim, els dos motors, connectats al pont en H.

El pont en H, l'ultrasó i el GPS, estan controlats per l'Arduino mitjançant una connexió amb els pins digitals.

També apareix un bumper, muntat amb un divisor de tensió mitjançant una resistència d'1 kΩ.

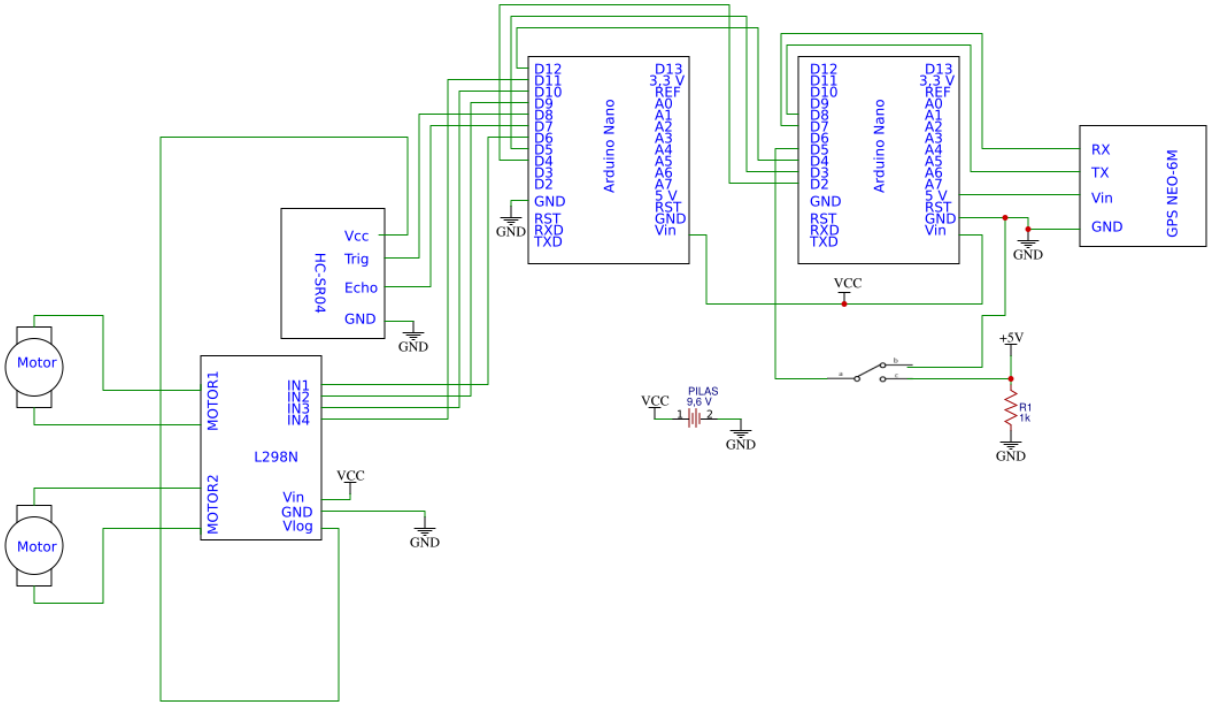

Figura 34. Connexionat Iron.

6.1.4. Diagrama de flux

En els següents diagrames s'explicarà el funcionament del primer robot.

6.1.4.1. Funció Loop

Funció bucle, encarregada de seqüenciar les funcions de tot el programa y variar la velocitat dels robots depenent de la proximitat dels objectes ("Figura 35").

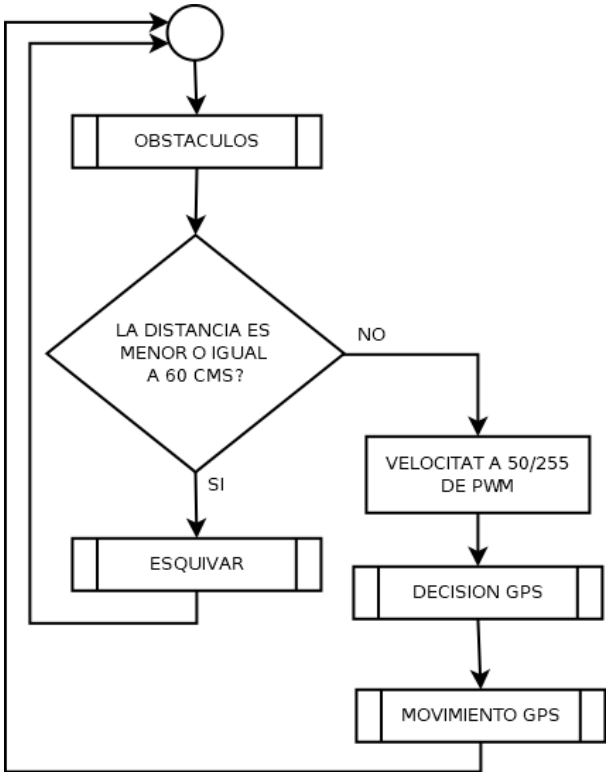

Figura 35. Diagrama de flux, funció "loop".

6.1.4.2. Funció Esquivar

La funció esquivar ("Figurta 36"), és la funció encarregada de fer la correcció de la trajectòria quan el primer robot es troba amb un impediment en el seu camí. Primer s'allunyarà de l'objecte per després procedir a esquivar-lo per l'esquerre.

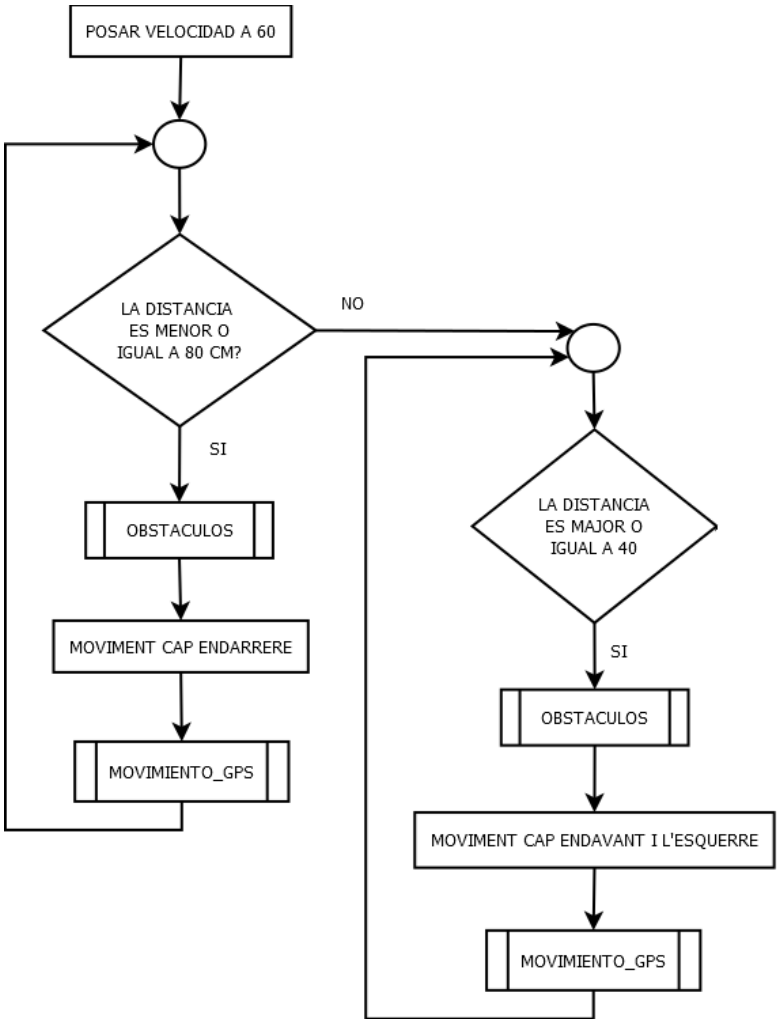

Figura 36. Diagrama de flux, funció "esquivar".

6.1.4.3. Funció Decision_GPS

En el següent diagrama ("Figura 37") s'explica el funcionament de la funció decision_GPS. Aquesta funció és l'encarregada de donar-li a la variable salida un valor numèric decimal depenent del valor binari que la placa del GPS li aportarà. Amb aquesta variable el robot sabrà cap a on s'ha de moure en cada moment.

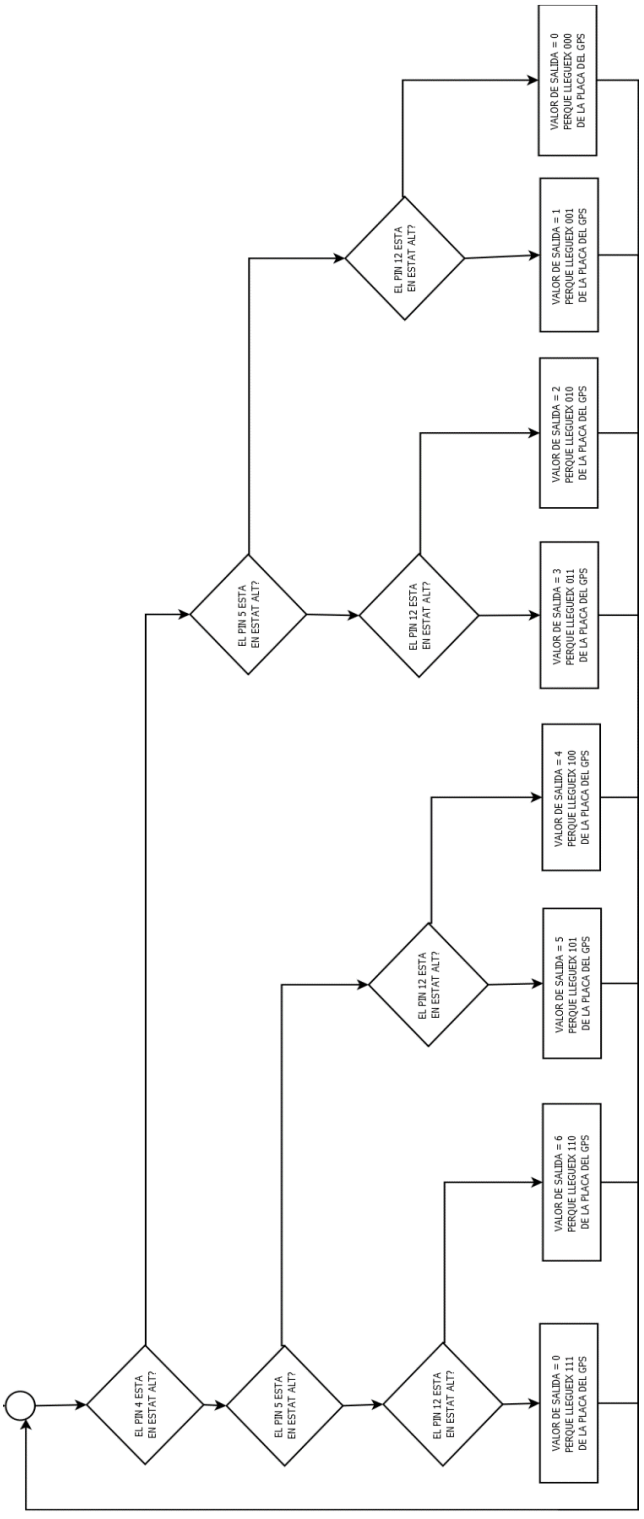

Figura 37. Diagrama de flux, funció "decisión_GPS".

6.1.4.4. Funció Obstaculos

Funció encarregada de realitzar una mediana de tres valors llegits per l'ultrasò ("Figura 38"), d'aquesta manera es pot saber en cada moment si el robot està a prop d'algun obstacle.

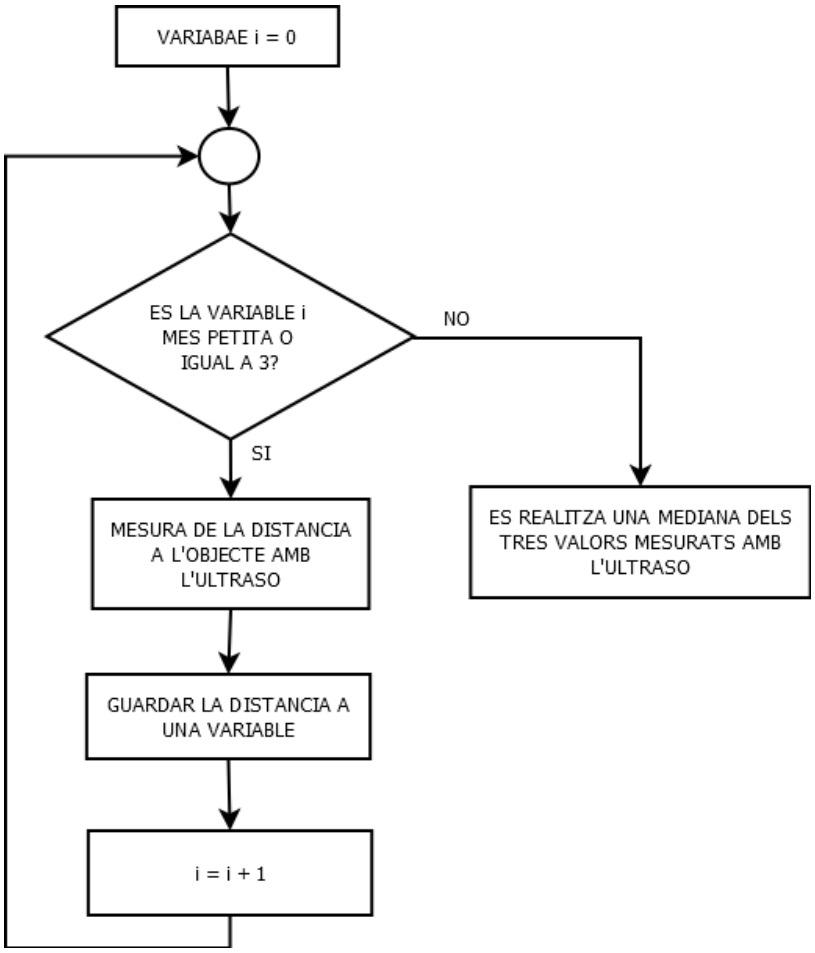

Figura 38. Diagrama de flux, funció "obstaculos".

6.1.4.5. Funció Movimiento_GPS

En aquesta funció ("Figura 39"), el programa accionarà els motors en un sentit, en un altre o els pararà depenent del valor que adquireixi la variable salida en qualsevol de les funcions anteriors. Aquesta activació es realitzarà mitjançant el pont en H.

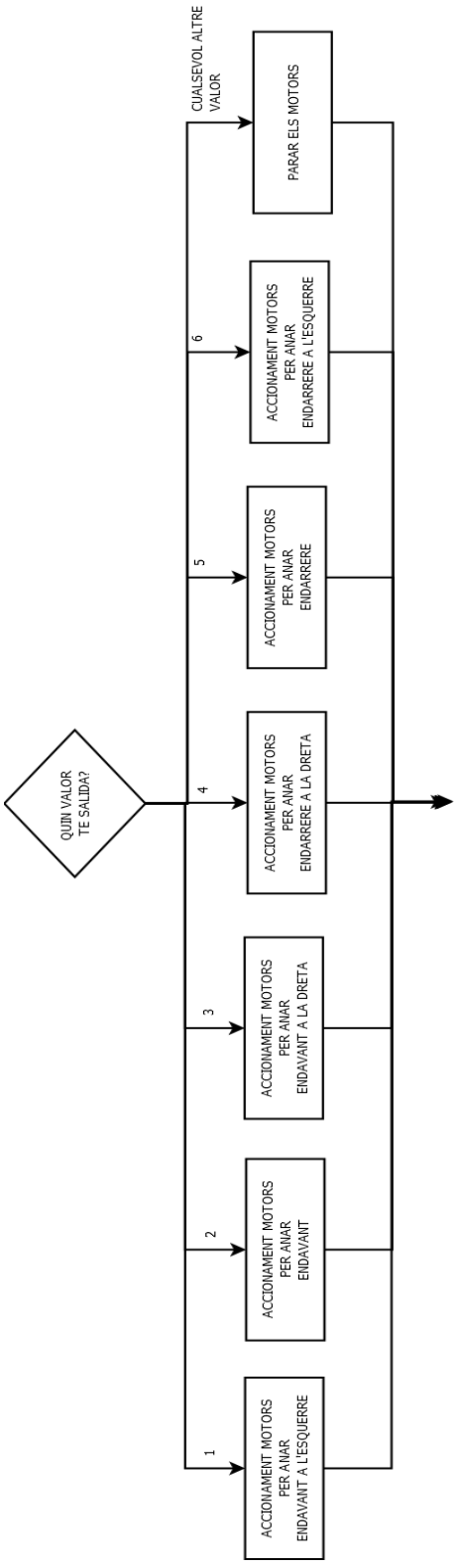

Figura 39. Diagrama de flux, funció "movimiento_GPS".

6.2. Groot

La funció del Groot (“Figura 40”) consisteix a col·locar la llavor al forat que ha realitzat l’Iron.

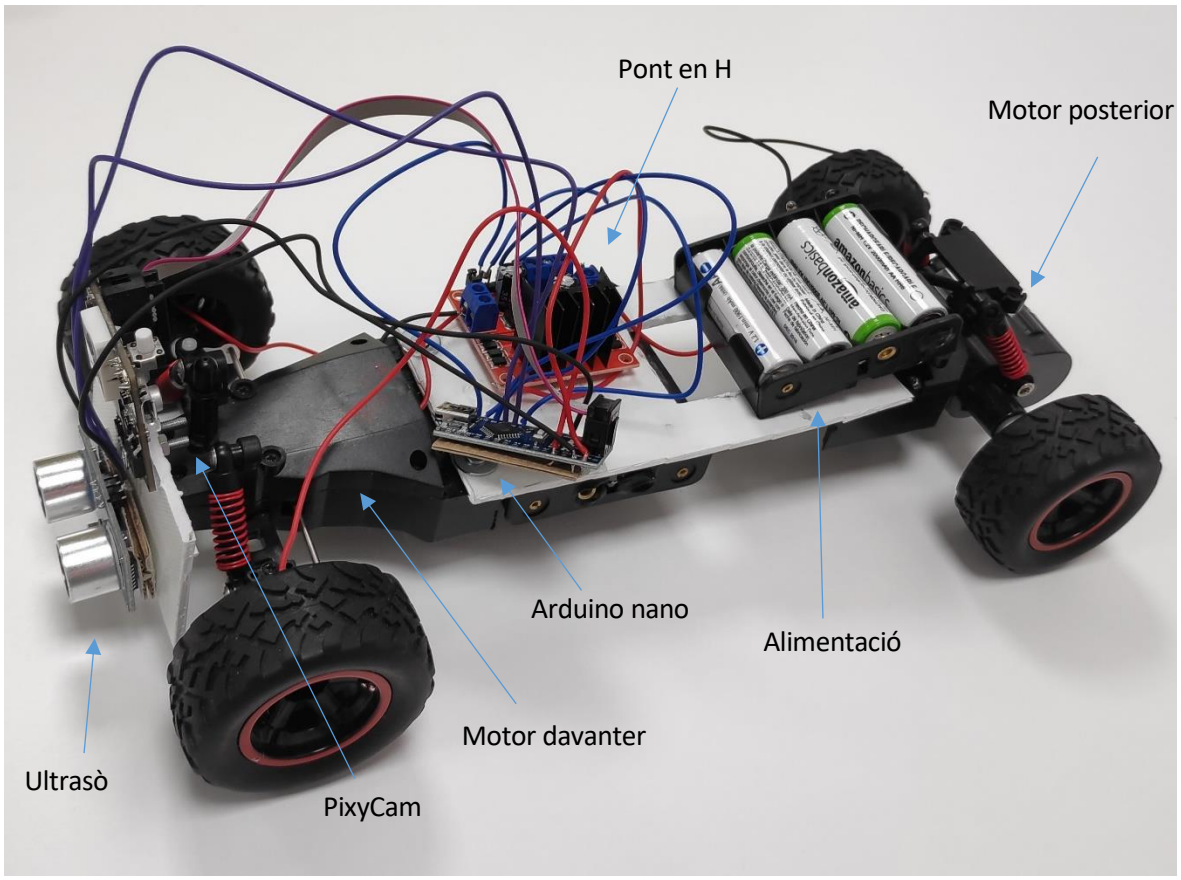

Figura 40. Imatge primer robot

6.2.1. Mecànica

L’útil d’aquest robot consisteix en un recipient on portarà totes les llavors. Just a la sortida d’aquest recipient hi haurà una esfera enganxada a un motor que la farà girar. A la pilota hi haurà un forat de les mateixes dimensions que una llavor per poder deixar caure únicamanet una.

6.2.2. Electrònica

Està formada per una càmera, que és l’encarregada de fer el seguiment del primer robot, un pont en h, que és l’encarregat de fer girar els motors del cotxe en diferents direccions per tal de poder tindre a l’abast tots els moviments necessaris, i per últim dos sensors de seguretat per tal de no xocar amb cap objecte o animal a l’hora de reforestar, aquests sensors són un d’ultrasons i un bumper, aquest últim

és l'encarregat de parar el cotxe si actua, ja que significaria que alguna cosa no està funcionant correctament perquè els altres sensors no han detectat.

6.2.3. Connexionat

En la següent imatge ("Figura 41") es pot apreciar l'esquemàtic del segon dels tres robots. La placa Arduino, alimentada amb 9.8 V, igual que el pont en H. Aquest robot també consta d'un ultrasò, alimentat amb 5 V extrets de l'Arduino i la pixyCam, connectada amb els pins ICSP, que l'alimenten i la comuniquen amb l'Arduino. Per últim, els dos motors, connectats al pont en H.

El pont en H i l'ultrasò, estan controlats per l'Arduino mitjançant una connexió amb els pins digitals.

També apareix un bumper, muntat amb un divisor de tensió mitjançant una resistència d'1 kΩ.

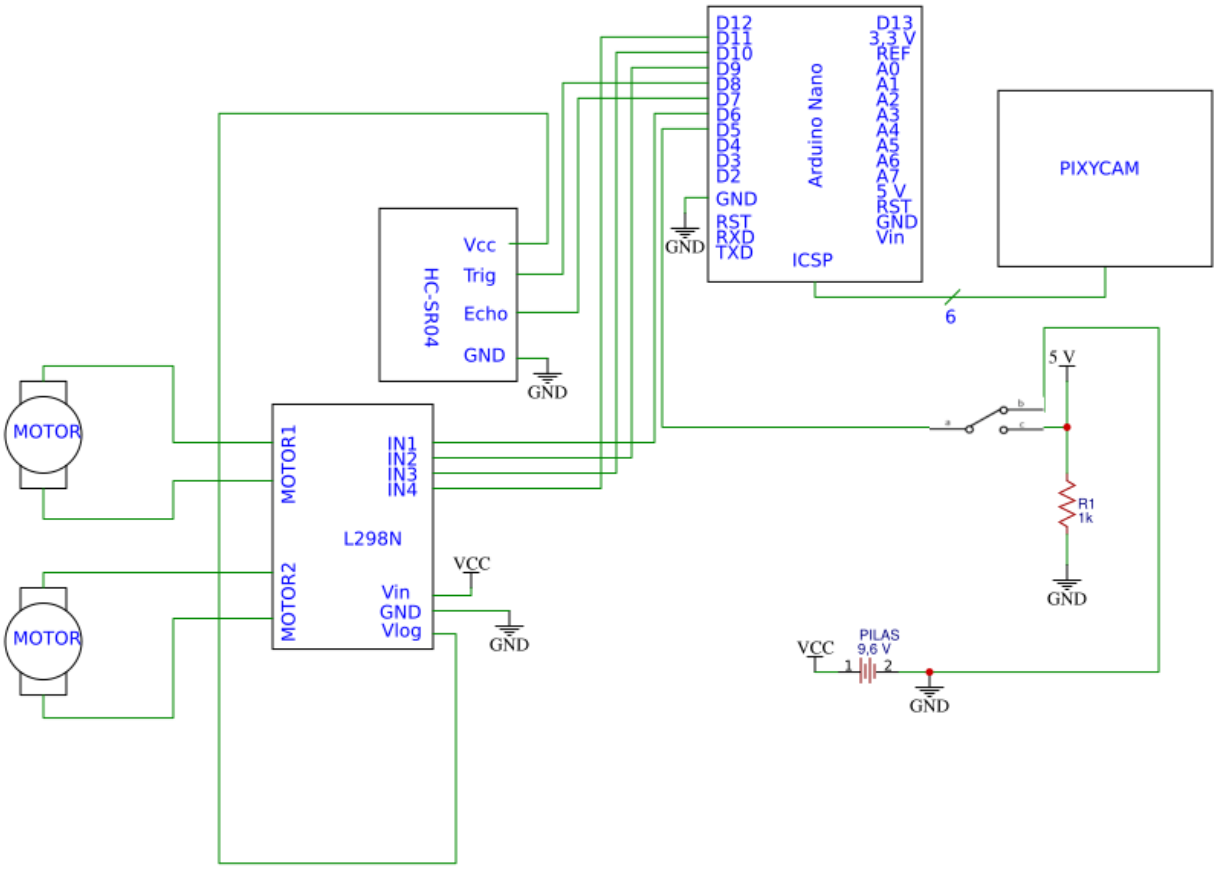

Figura 41. Connexionat groot.

6.2.4. Diagrama de flux

En els següents diagrames s'explicarà el funcionament del primer robot.

6.2.4.1. Funció Loop

Funció bucle, encarregada de seqüenciar les funcions de tot el programa (“Figura 42”), una darrere de l’altre, primer entrar a la funció càmera, després obstaculo, tot seguit correguir i per últim movimento_GPS, funcions que s’explicaran en els següents punts.

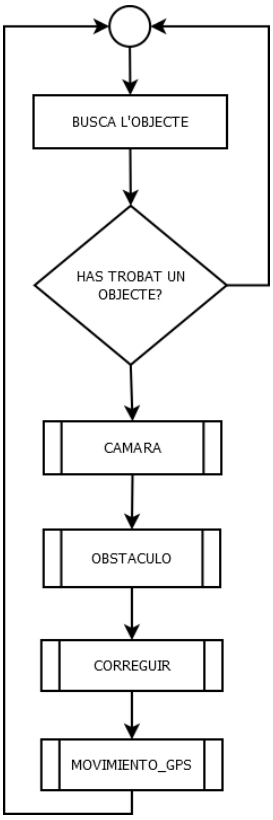

Figura 42. Diagrama de flux, funció "loop".

6.2.4.2. Funció Camara

Funció encarregada de governar el moviment del segon robot depenent la posició X del primer robot (“Figura 43”).

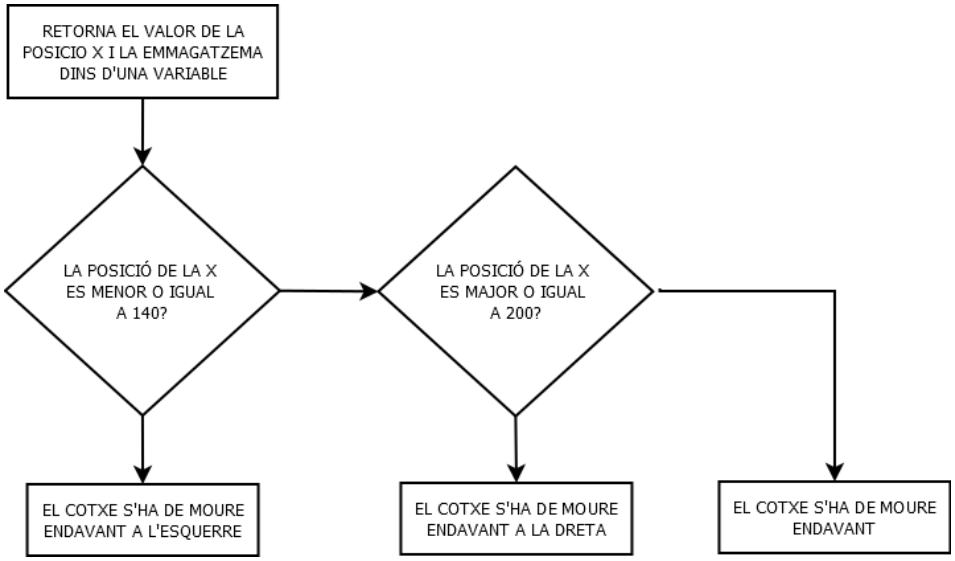

Figura 43. Diagrama de flux, funció "camara".

6.2.4.3. Funció Obstaculo

Funció encarregada de realitzar una mediana de tres valors llegits per l'ultrasò ("Figura 44"), d'aquesta manera es pot saber en cada moment si el robot està a prop d'algun obstacle.

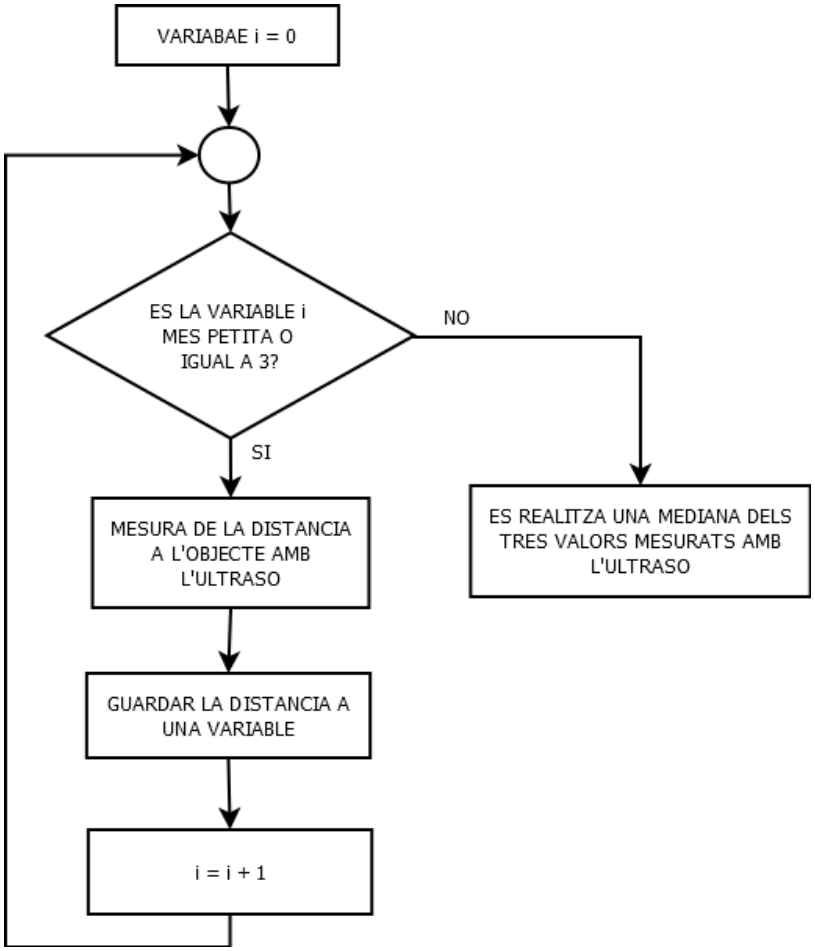

Figura 44.Diagrama de flux, funció "obstaculo".

6.2.4.4. Funció Correguir

Aquesta funció (“Figura 45”) és l’encarregada de corregir la velocitat o direcció del robot depenent de la proximitat al robot de davant.

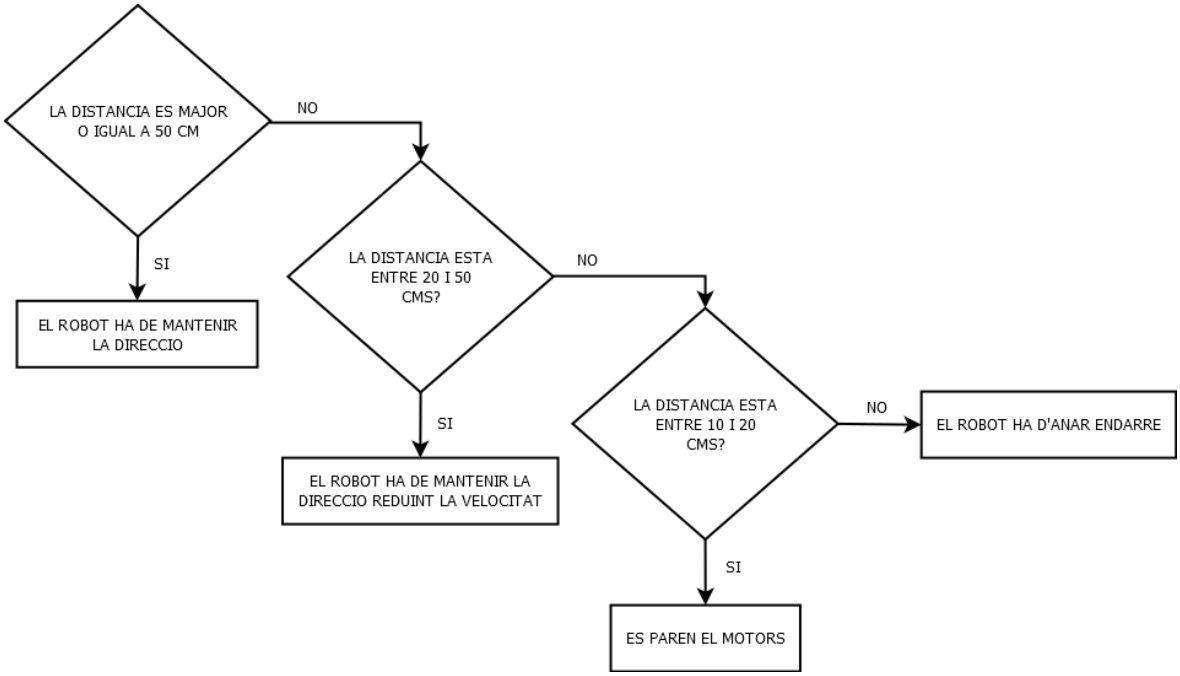

Figura 45. Diagrama de flux, funció "Corregir".

6.2.4.5. Funció Movimineto_GPS

En aquesta funció ("Figura 46"), el programa accionarà els motors en un sentit, en un altre o els pararà depenent del valor que adquireixi la variable salida en qualsevol de les funcions anteriors. Aquesta activació es realitzarà mitjançant el pont en H.

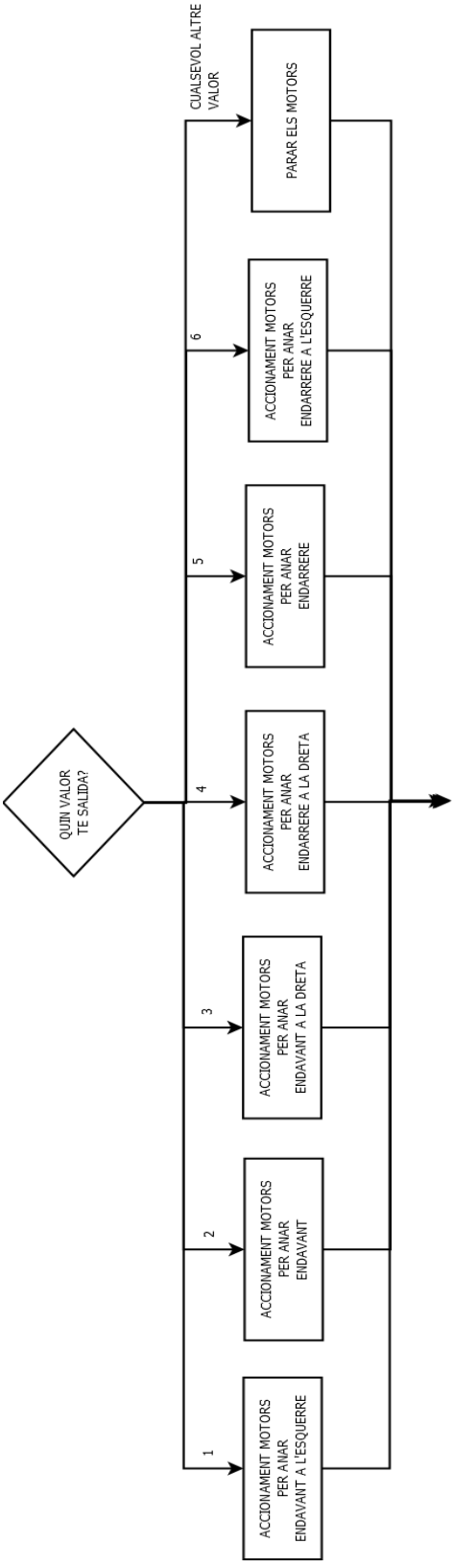

Figura 46. Diagrama de flux, funció "Movimiento_GPS".

6.3. Wall

La seva funció consisteix a tapar el forat fet pel primer robot amb la llavor dintre deixada pel segon (“Figura 47”).

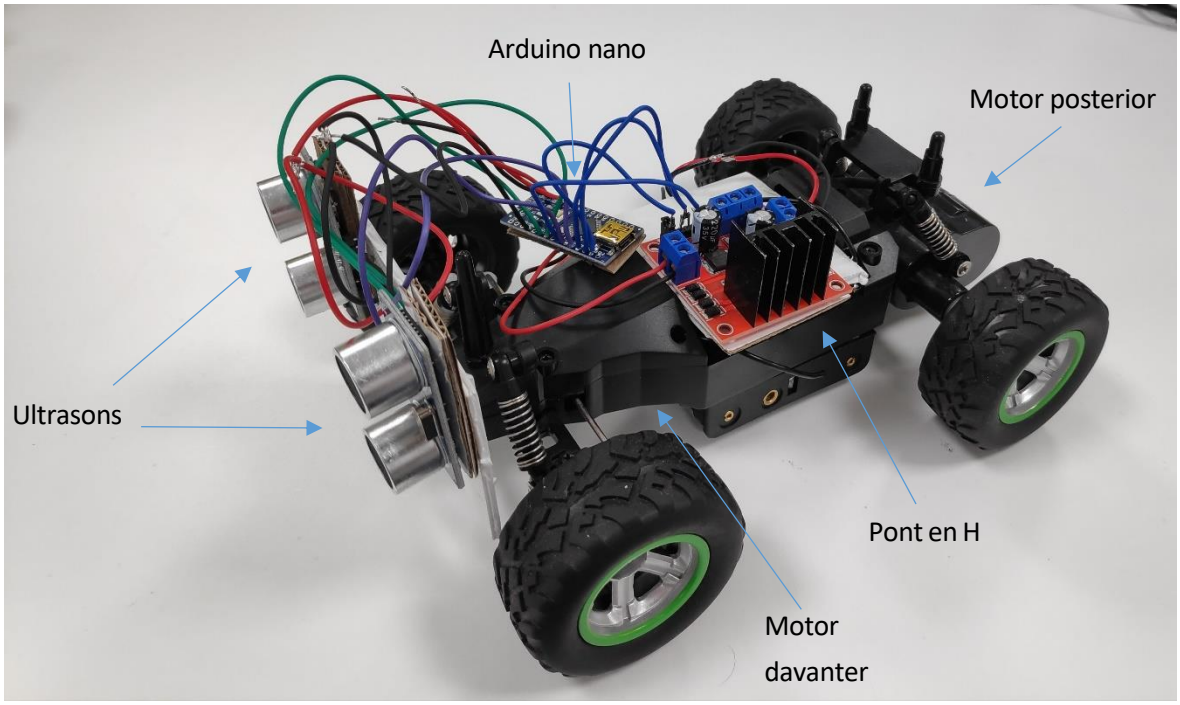

Figura 47. Imatge tercer robot

6.3.1. Mecànica

Per tapar els forats, s’ha optat per instal·lar a la part davantera del robot una pala que empenyerà tota la sorra moguda pel primer robot a l’hora de fer el forat.

6.3.2. Electrònica

El seguiment del segon rebot es realitza amb dos ultrasons, depenent de si aquests estan detectant o no, un pont en h, que és l’encarregat de fer girar els motors del cotxe en diferents direccions per tal de poder tindre a l’abast tots els moviments necessaris, i per últim dos sensors de seguretat per tal de no xocar amb cap objecte o animal a l’hora de reforestar, aquests sensors són dos ultrasons i un bumper, aquest últim és l’encarregat de parar el cotxe si actua, ja que significaria que alguna cosa no està funcionant correctament perquè els altres sensors no han detectat.

6.3.3. Connexionat

En la següent imatge (“Figura 48”) es pot apreciar l’esquemàtic del tercer dels tres robots. La placa Arduino, alimentada amb 9.8 V, igual que el pont en H. Aquest robot també consta de dos ultrasons, alimentats amb 5 V extrets de l’Arduino. Per últim, els dos motors, connectats al pont en H.

El pont en H i els ultrasons, estan controlats per l’Arduino mitjançant una connexió amb els pins digitals.

També apareix un bumper, muntat amb un divisor de tensió mitjançant una resistència d’1 kΩ.

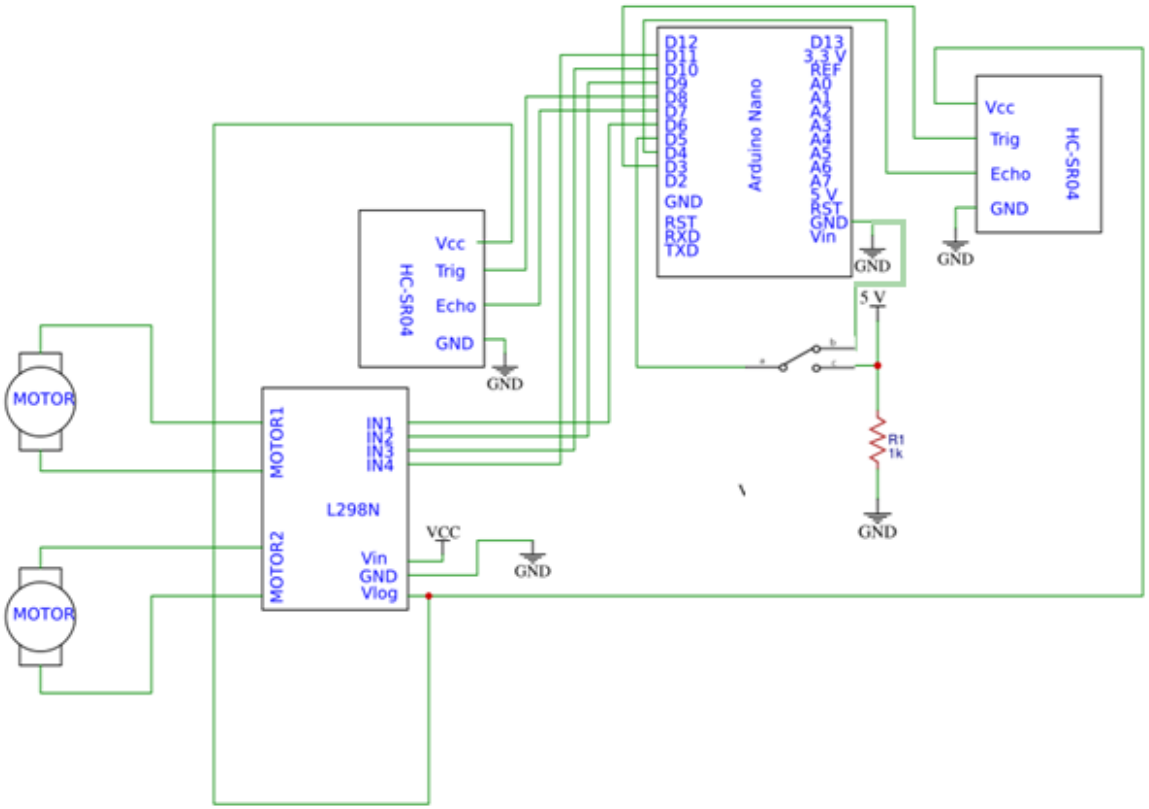

Figura 48. Connexionat Wall

6.3.4. Diagrama de flux

En els següents diagrames s’explicarà el funcionament del primer robot.

6.3.4.1. Loop

Funció bucle, encarregada de seqüenciar les funcions de tot el programa (“Figura 49”), una darrere de l’altre, primer entrar a la funció càmera, després obstaculo, tot seguit correguir i per últim movimento_GPS, funcions que s’explicaran en els següents punts.

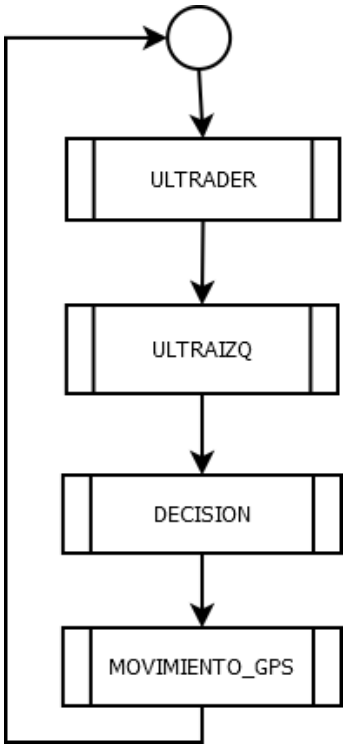

Figura 49. Diagrama de flux, funció "Loop".

6.3.4.2. UltraDer

Funció encarregada de realitzar una mediana de tres valors llegits per l’ultrasò (“Figura 50”), d’aquesta manera es pot saber en cada moment si el robot està a prop d’algun obstacle.

En el cas d’aquest robot també servirà per fer el seguiment del robot que porta davant, el segon.

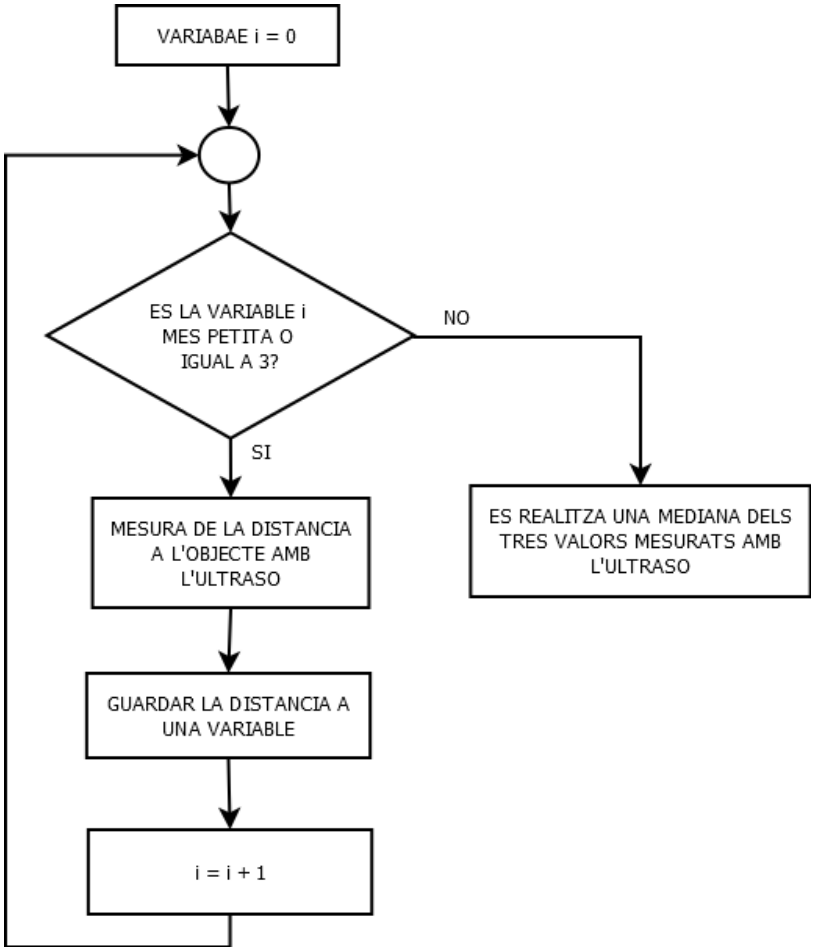

Figura 50.Diagrama de flux, funció "UltraDer".

6.3.4.3. UltraIzq

Funció encarregada de realitzar una mediana de tres valors llegits per l'ultrasò ("Figura 51"), d'aquesta manera es pot saber en cada moment si el robot està a prop d'algun obstacle.

En el cas d'aquest robot també servirà per fer el seguiment del robot que porta davant, el segon.

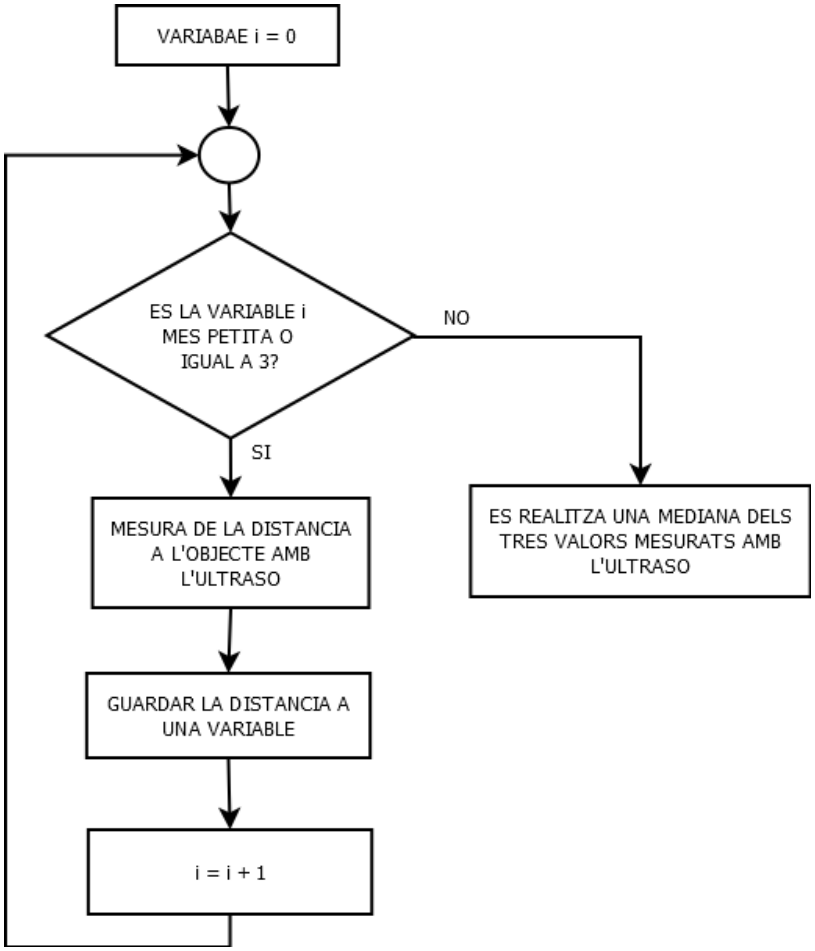

Figura 51.Diagrama de flux, funció "UltraIzq".

6.3.4.4. Decision

A la següent imatge (“Figura 52”) es mostra el comportament del codi de la presa de decisió que ha de realitzar el robot depenent de la distància que llegeixi cada ultrasò. Si l’ultrasò esquerre mesura molta distància vol dir que el robot de davant s’està allunyant per l’esquerre, això significa que està girant a la dreta i viceversa.

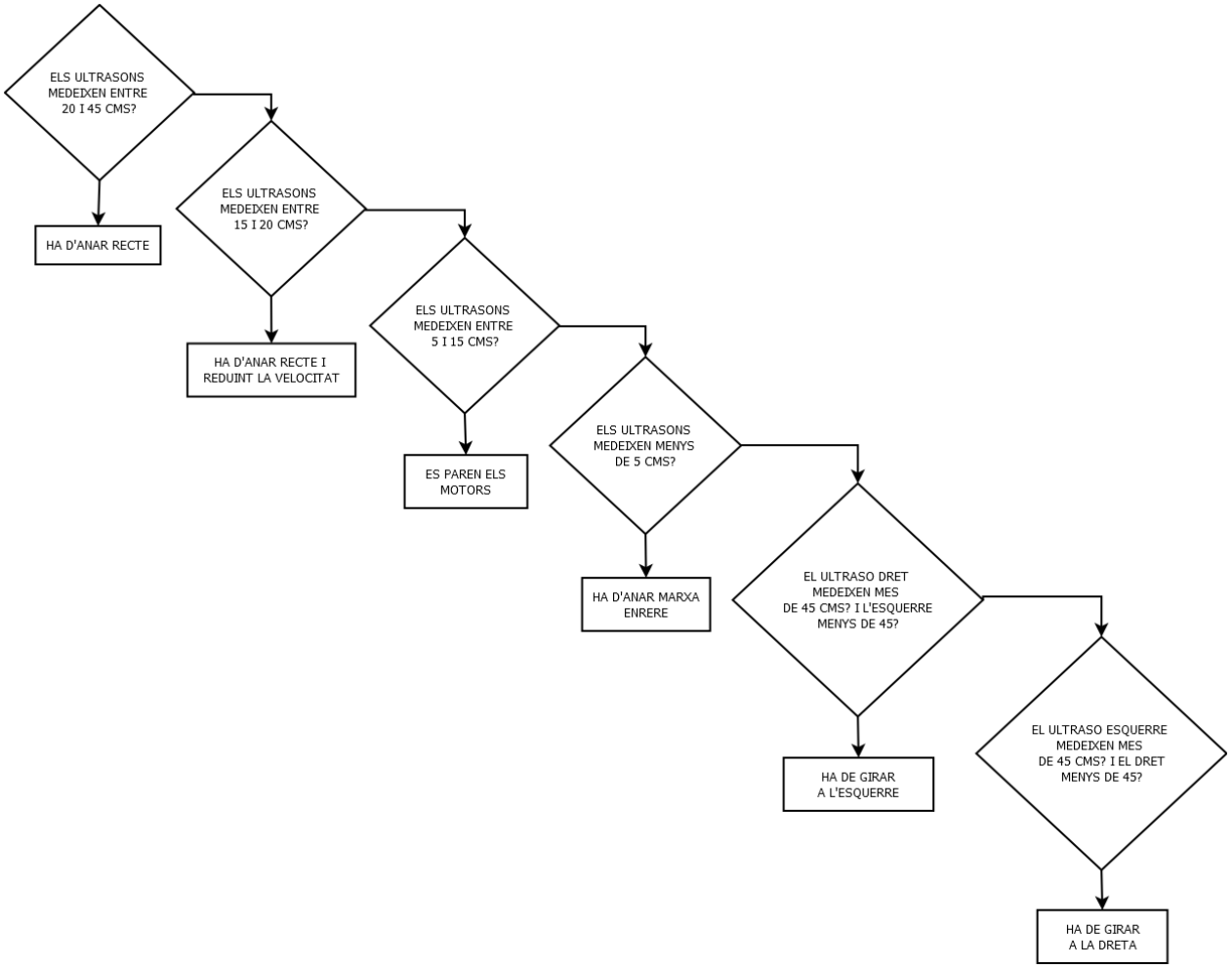

Figura 52. Diagrama de flux, funció "Decisión".

6.3.4.5. Movimiento_GPS

En aquesta funció (“Figura 53”), el programa accionarà els motors en un sentit, en un altre o els pararà depenent del valor que adquireixi la variable salida en qualsevol de les funcions anteriors. Aquesta activació es realitzarà mitjançant el pont en H.

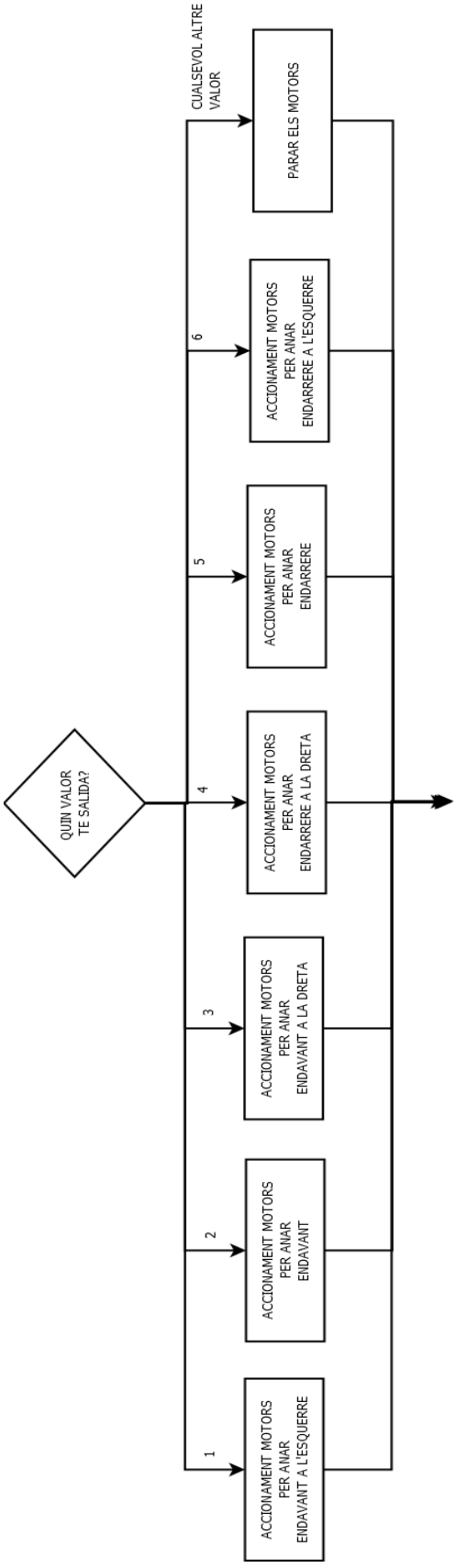

Figura 53.Diagrama de flux, funció "Movimiento_GPS".

6.4. Autonomia

En la taula següent es mostren les autonomies dels 3 robots.

Taula 3. Autonomia dels robots.

Robot	Autonomia
Iron	2 h
Groot	1 h 40 min
Wall	2 h

S'observa que el que menor autonomia té es Groot. La diferencia es deu a la camara, la qual consumeix una quantitat d'energia més elevada que la resta de components de control i sensat.

Les mesures mostrades aquí s'han obtingut de manera experimental.

7. Anàlisi de l'impacte ambiental

A causa de la situació mediambiental actual, avui en dia és necessari pensar en l'impacte ambiental de qualsevol projecte. Aquest treball no és cap excepció, per la qual cosa es valoraran tres apartats principalment: materials emprats, energia i finalitat del treball.

7.1.1. Materials emprats

Per a la realització del treball va ser de molta ajuda l'autorització per obtenir materials del punt verd d'Esplugues de Llobregat. D'aquest punt verd s'han aconseguit alguns materials, com motors, xapes de plàstic o d'alumini emprats en la modificació dels cotxes radio control. Així mateix s'ha intentat reciclar cotxes de radio control antics, no obstant això, com no ha sigut possible aconseguir suficients, s'han hagut de comprar.

Respecte als components electrònics, tots compleixen normativa RoHS i han hagut de ser comprats.

7.1.2. Energia

Per a l'alimentació dels robots s'ha optat per piles de gran rendiment. Permeten 2100 càrregues i perden molt poca càrrega amb el pas del temps. Això suposa un estalvi energètic i en materials important.

7.1.3. Finalitat del treball

La finalitat d'aquest treball és la reforestació, tenint en compte la situació critica que viu el nostre món, aquest és possiblement el punt més positiu en l'impacte ambiental. Iniciatives d'algunes comunitats per recuperar boscos de camps agrícoles o plantacions massives com les vistes a la Xina o Etiòpia fan que projectes com aquest siguin molt útils. Encara més palpable després de l'article publicat a la revista Science d'un equip d'investigadors liderat per l'institut de biologia integrativa de l'institut federal suís de tecnologia de Zuric, on queda patent que plantar arbres seria la solució més eficaç, barata i rapida de combatre el canvi climàtic.

En aquest cas es tracta de robots de baix cost que poden ajudar a aquest tipus d'iniciatives.

Conclusions

Amb la realització d'aquest treball s'ha intentat dissenyar i fabricar un procés automàtic de reforestació. Desgraciadament per motius de temps no ha estat possible finalitzar tot el projecte. Al moment d'entregar aquest treball s'ha arribat al punt de tenir els 3 robots dissenyats i construïts comprovant el seu funcionament experimentalment.

Durant el transcurs del treball s'ha hagut d'utilitzar els coneixements adquirits durant els nostres estudis i de realitzar probes amb components amb els que no s'havien treballat, tals com infrarojos, ponts H, bumpers, Arduino... Mitjançant l'estudi previ i la realització de les proves s'han escollit uns sensors o uns altres per a cada robot. Per exemple, després de fer les proves es va descartar el sensor infraroig per ser massa inestable. Tampoc s'han pogut instal·lar els bumpers per falta de temps, tot hi que el codi esta implementat i provat, està comentat per tal de que no interfereixi.

Per altre banda, considerem la realització del treball en grup una bona experiència, i molt útil per poder avançar la feina diària que comporta la realització d'aquest treball.

Possibles millores

Finalment proposar algunes possibles millores a realitzar en aquest treball:

- Separar l'alimentació dels motors de l'electrònica: Tot hi que l'electrònica s'alimenti mitjançant un regulador, seria una bona idea que tingués la seva pròpia bateria per acabar de protegir-la davant els pics de potencia que produeixen els motors.
- Erugues: Inicialment es van descartar per ser poc econòmiques, però un cop avançat el treball s'ha vist que haguessin estat una bona inversió, ja que el sistema actual només permet realitzar uns girs amples.
- Busca una combinació de sensors per detectar obstacles: Durant el treball s'han buscat més alternatives, però l'única que s'ha mostrat fiable a estat la tecnologia d'ultrasons. Buscar algun altre que la complementes per detectar obstacles podria ser molt positiu per al projecte.

Anàlisi Econòmic

En aquest apartat es detalla el pressupost del treball, diferenciant principalment dos grans blocs: Preu dels materials emprats i hores emprades.

Taula 3. Anàlisi econòmic

Concepte	Preu unitari	Quantitat	Total
Hores emprades	25,00 €/h	1200 h	30.000€
Arduino nano	3,196 €	7	22,37 €
Cotxes teledirgits	20,00 €	4	80 €
GPS	14,99 €	2	29,98 €
Rodament	6,05 €	1	6,05 €
PixyCam	65,00 €	1	65,00 €
Ultrasons	1,99 €	10	19,90 €
Infraroig	12.97 €	2	25,94 €
Càmera	12,09 €	1	12,09 €
Portapiles	1,60 €	5	7,99 €
Pont en H	2,80 €	10	28,00 €
Bumpers	0,12 €	10	1,20 €
TOTAL			**30.298,52 €**

Es pot observar com alguns materials estan quantificats per sobre de les unitats finals emprades, això es deu a que durant el projecte alguns chips es van trencar i d'altres no es van arribar a utilitzar, però venien en pack.

Bibliografia

"Wiki:V2:Start [Documentation]." Accessed September 26, 2019. https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:start.

Barbera, Javier Colomer. "TRABAJO FINAL DE MÁSTER Estudio de Los Sensores Para La Detección de Obstáculos Aplicables a Robots Móviles," n.d.

Mayné, Jordi. "Sensores Acondicionadores y Procesadores de Señal," n.d.

"Sensores y Actuadores," n.d.

"CAPITULO VII SENSORES," n.d.

"2. SENSORES," n.d.

Client, Cher. "Manuel d'utilisation Du Capteur à Ultrasons VMA306." Accessed September 26, 2019. www.gotronic.fr.

"EasyEDA – Simulador de Circuitos y Diseño de Circuitos Impresos Online." Accessed September 26, 2019. https://easyeda.com/es.

"Tutorial Uso Driver L298N Para Motores DC y Paso a Paso Con Arduino." Accessed September 26, 2019. https://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/.

"Módulo Controlador de Motores L298N | Tienda y Tutoriales Arduino." Accessed September 26, 2019. https://www.prometec.net/l298n/.

"Controlar Motores de Corriente Continua Con Arduino y L298N." Accessed September 26, 2019. https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/.

"El Puente H: Invirtiendo El Sentido de Giro de Un Motor Con Arduino." Accessed September 26, 2019. http://panamahitek.com/el-puente-h-invirtiendo-el-sentido-de-giro-de-un-motor-con-arduino/.

"≫ SR04 ▷ Medir Distancia Por Ultrasonido Con Arduino | PatagoniaTec." Accessed September 26, 2019. https://saber.patagoniatec.com/2014/10/tutorial-modulo-ultrasonico-arduino-argentina-ptec-sr04-srf05-us020/.

"Medir Distancia Módulos SR04 SRF05 Ultrasonidos Arduino." Accessed September 26, 2019. https://polaridad.es/medir-distancias-ultrasonidos-sr04-srf05-arduino/.

"SRF05 SENSOR DISTANCIAS ULTRASONIDOS SIMPLE." Accessed September 26, 2019. http://www.superrobotica.com/s320111.htm.

"Medir Distancia Con Arduino y Sensor de Ultrasonidos HC-SR04." Accessed September 26, 2019. https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/.

"GitHub - JRodrigoTech/Ultrasonic-HC-SR04: Ultrasonic HC-SR04 Library for Arduino IDE." Accessed September 26, 2019. https://github.com/JRodrigoTech/Ultrasonic-HC-SR04.

"GitHub - Chibike/Arduino-Ultrasonic: Arduino HC-SR05 Ultrasonic Library." Accessed September 26, 2019. https://github.com/chibike/Arduino-Ultrasonic.

"Velleman HC-SR05 Ultrasound Sensor Not Measuring beyond 5 Cm." Accessed September 26, 2019. https://forum.arduino.cc/index.php?topic=534259.0.

"Velleman HC-SR05 Ultrasound Sensor Not Measuring beyond 5 Cm." Accessed September 26, 2019. https://forum.arduino.cc/index.php?topic=534259.0.

"Modulación de Señal Infrarroja." Accessed September 26, 2019. http://www.tutoelectro.com/tutoriales/electronica-basica/modelado-de-senal-infrarroja/.

"Ventajas de Utilizar Bombas de Carga En El Diseño de Circuitos – ElectronicosOnline.Com Magazine." Accessed September 26, 2019. https://www.electronicosonline.com/ventajas-de-utilizar-bombas-de-carga-en-el-diseno-de-circuitos/.

"Obtener Voltajes Superiores Al de Alimentación o Voltajes Negativos – TallerElectronica.Com / Blog." Accessed September 26, 2019. https://tallerelectronica.com/2016/07/01/conseguir-voltajes-superiores-al-de-alimentacion-o-voltajes-negativos/.

"Bibliografia - Documentos de Google." Accessed September 26, 2019. https://docs.google.com/document/d/1IOM9JLY62koJHsDGUGnsbHWMcgSDuc6NDWIyic82VzI/edit.

"SB-Projects - IR Index." Accessed September 26, 2019. https://www.sbprojects.net/knowledge/ir/index.php.

"Localización GPS Con Arduino y Los Módulos GPS NEO-6." Accessed September 26, 2019. https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/.

"Tutorial Sensor de Distancia SHARP." Accessed September 26, 2019. https://naylampmechatronics.com/blog/55_tutorial-sensor-de-distancia-sharp.html.

"Arduino Reference." Accessed September 26, 2019. https://www.arduino.cc/reference/en/#variables.

"I2C Cómo Conectar Dos Arduino Mediante Este Protocolo." Accessed September 26, 2019. https://programarfacil.com/blog/arduino-blog/conectar-dos-arduinos-i2c/#Conectar_dos_Arduinos.

"3 Formas de Cultivar Pinos - WikiHow." Accessed September 26, 2019. https://es.wikihow.com/cultivar-pinos.

Annex A

A continuació es presenten el codi de les quatre plaques que apareixen en el projecte, primer la placa que controla el robot 1, després la que controla el robot 2, tot seguit la que controla l'últim robot i per acabar la placa que controla el GPS del primer robot.

A1. Codi del primer robot (Iron)

```
// Codi TFG Marc Guarnido Arquero i Sergio Ordiaz Bonet. Robot 1
"IronMan" par a automatizacio de processos forestals.

//Librerias
#include "QuickMedianLib.h"   //Mediana

#include <TimerOne.h>   //Timer

//Pins Ultrasons
#define ECHOPIN 8   // Pin de lectura de la senyal de l'ultraso
#define TRIGPIN 7   // Pin de enviament de la senyal de l'ultraso

//Pins GPS
#define GPS1pin 4   //Correspon a sal pin numero 1 de l'altre arduino
#define GPS2pin 5   //Correspon a sal pin numero 2 de l'altre arduino
#define GPS3pin 12    //Correspon a sal pin numero 3 de l'altre arduino

//Pins pont en H
#define pinIN1A 11   //Pin per anar cap a darrere (acciona motor de
darrere)
#define pinIN2A 10  //Pin per anar cap endavant (acciona motor de
darrere)
#define pinIN1B 9    //Pin per anar cap a la dreta (acciona motor de
davant)
#define pinIN2B 6    //Pin per anar cap a l'esquerre (acciona motor de
davant)

//Variables funcion obstaculos
int distancia_sonido[3];   //Variable que emmagatzema 3 valors obtinguts
per l'ultraso
int med_ultra;    //Mediana dels 11 valors emmagatzemats anteriorment per
l'ultraso

//Variable PWM
int speed = 50;    //Variable que fara augmentar o disminuir la velocitat
del robot depenent de les condicions

//Variable sortida GPS
int salida;   //Depenent del valor que adopti el robot es moura cap un
lloc o cap a un altre
int distancia;

//Variable para esquivar
bool evitar = false;    //Variable que fa entrar a la funcio esquivar
```

```
//*****************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//*****************************************************************//

void setup() {
  Serial.begin(9600); //

  //PINs ultrasonidos
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);


  //PINs puente en H
  pinMode(pinIN1A, OUTPUT);
  pinMode(pinIN2A, OUTPUT);
  pinMode(pinIN1B, OUTPUT);
  pinMode(pinIN2B, OUTPUT);
}

//*****************************************************************//
// Nom de la funcio: movimiento_GPS()
// Funcio: Funcio encarregada d'activar els motors per direccionar el
robot amb velocitat variable.
//*****************************************************************//

void movimiento_GPS() {
  switch (salida) {
    case 1:   //Anar cap a dalt i cap a l'esquerre
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;

    case 2:   //Anar cap a dalt
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;

    case 3:   //Anar cap a dalt i cap a la dreta
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 4:   //Anar cap a baix i cap a la dreta
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 5:   //Anar cap a baix
```

```
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;

    case 6:   //Anar cap a baix i cap a l'esquerre
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;

    default:    //El estat default fa que els motors es quedin totalment
quiets
      digitalWrite(pinIN1A, LOW);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;
  }
}

//*********************************************************************//
// Nom de la funcio: decision_GPS()
// Funcio: Funcio que interpreta la comunicacio paralel que rep de la
placa conectada al GPS i la tradueix a una variable usada a la resta del
codi
//*********************************************************************//

void decision_GPS() {
  if (digitalRead(4) == HIGH) {
    if (digitalRead(5) == HIGH) {
      if (digitalRead(12) == HIGH) salida = 0; //Esta rebent 111 de la
placa del GPS, el que significa que la variable sortida ha de ser 0
      else salida = 6; //Esta rebent 110 de la placa del GPS, el que
significa que la variable sortida ha de ser 6
    }
    else {
      if (digitalRead(12) == HIGH) salida = 5; //Esta rebent 101 de la
placa del GPS, el que significa que la variable sortida ha de ser 5
      else salida = 4; //Esta rebent 100 de la placa del GPS, el que
significa que la variable sortida ha de ser 4
    }
  }
  else {
    if (digitalRead(5) == HIGH) {
      if (digitalRead(12) == HIGH) salida = 3; //Esta rebent 011 de la
placa del GPS, el que significa que la variable sortida ha de ser 3
      else salida = 2; //Esta rebent 010 de la placa del GPS, el que
significa que la variable sortida ha de ser 2
    }
    else {
      if (digitalRead(12) == HIGH) salida = 1; //Esta rebent 001 de la
placa del GPS, el que significa que la variable sortida ha de ser 6
      else salida = 0; //Esta rebent 000 de la placa del GPS, el que
significa que la variable sortida ha de ser 0 (default)
    }
```

```
  }
}

//*********************************************************************//
// Nom de la funcio: obstaculos()
// Funcio: Funcio que calcula la distancia a la que esta el robot dels
objectes de devant (es fan medianas per tal de tindre valors mes
estables)
//*********************************************************************//

void obstaculos() {
  //Ultraso
  int i = 0;
  while (i <= 3) {
    digitalWrite(TRIGPIN, LOW);   //Pin de dispar a esta baix
    delayMicroseconds(5);
    digitalWrite(TRIGPIN, HIGH);    //Envia un estat alt per tal de saber
on es l'objecte
    delayMicroseconds(15);
    digitalWrite(TRIGPIN, LOW);   //Es torna a posar el valor del trigger
a 0
    distancia = pulseIn(ECHOPIN, HIGH);   //Llegueix el temps que ha
tardat en retornar el estat alt enviat anteriorment
    distancia = distancia / 58;   //Calcula la distancia del pols
(anteriorment es mesurava en temps)
    if (distancia>0){
      distancia_sonido[i] = distancia;    //S'emmagatzema 3 valors de
distancia
      i = i + 1;
    }
  }
  med_ultra = QuickMedian<int>::GetMedian(distancia_sonido,
3);    //Mitjancant la libreria es treu la mediana d'aquests 3 valors
mesurats anteriorment
}

//*********************************************************************//
// Nom de la funcio: esquivar()
// Funcio: Funcio encarregada de fer esquivar qualsevol objecte que es
trobi el robot per el cami
//*********************************************************************//

void esquivar() {
  speed = 45;   //Es canvia la variable de la valocitat a 45//255 de PWM
  while (med_ultra<=80){    //Mentre la distancia es menor o igual 80 cms
    obstaculos();   //Es torna a calcular la media de la distancia
    salida = 5;   //Marxa enrrere
    movimiento_GPS();   //Dona la ordre als motors
    }
  while (med_ultra >= 60){    //Mentre la distancia es superior o igual a
60 cms
    obstaculos();   //Es torna a calcular la media de la distancia
    salida = 1;   //Endavant i a l'esquerre per evitar l'obstacle
    movimiento_GPS();   //Dona la ordre als motors
  }
}

//*********************************************************************//
```

```
// Nom de la funcio: loop()
// Funcio: Funcio bucle, encarregada de sequenciar les funcions de tot el
programa y variar la velocitat dels robots depenent de la proximitat dels
objectes
//*********************************************************************//

void loop() {
  obstaculos();   //Entra a la funcio obstaculos perque el sensor mesuri
la distancia a la que esta l'objecte
  if ((med_ultra <= 80) && (med_ultra > 60)) {    //Si l'objecte esta
entre 20 y 30 centimetres, es redueix la velocitat com a mesure
preventiva
    speed = 50;   //La variable de la velocitat del motor es redueix
    decision_GPS();   //Es llegueix el que esta dient la placa del GPS
per saber cap a on s'ha de moure
    movimiento_GPS();   //Acciona els motors depenent de la variable
salida, aportada per el GPS
  }
  else if (med_ultra <= 60) {   //Si la distancia es menor a 20 cm, es
para el robot per esquivar aquest objecte
    esquivar();   //Entra a la funcio esquivar
  }
  else {    //Si no es detecta cap objecte el robot es moura de forma
normal
    speed = 60;    //La variable de la velicitat esta en el seu estat
nominal
    decision_GPS();   //Es llegueix el que esta dient la placa del GPS
per saber cap a on s'ha de moure
    movimiento_GPS();   //Acciona els motors depenent de la variable
salida, aportada per el GPS
  }
}
```

A2. Codi segon robot (Groot)

```
// Codi TFG Marc Guarnido Arquero i Sergio Ordiaz Bonet. Robot 2 "Groot"
par a automatizacio de processos forestals.

//Llibrerias

#include <TimerOne.h>
#include <QuickMedianLib.h>
#include <Pixy2.h>

//Pins Ultrasons
#define ECHOPIN 8   // Pin de lectura de la senyal de l'ultraso
#define TRIGPIN 7   // Pin de enviament de la senyal de l'ultraso

//Pins pont en H
#define pinIN1A 6   //Pin per anar cap a darrere (acciona motor de
darrere)
#define pinIN2A 9  //Pin per anar cap endavant (acciona motor de darrere)
#define pinIN1B 10    //Pin per anar cap a la dreta (acciona motor de
davant)
```

```
#define pinIN2B 3    //Pin per anar cap a l'esquerre (acciona motor de
davant)

// Variables de la camera
Pixy2 pixy;
int posX;

//Variables funcio obstaculos
int distancia_sonido[3];   //Variable que emmagatzema 11 valors obtinguts
per l'ultraso
int med_ultra;    //Mediana dels 11 valors emmagatzemats anteriorment per
l'ultraso
int distancia;

//Variable velocitat
int speed;

//Variables moviment
int salida;

//*************************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//*************************************************************************//

void setup()
{

  //PINs ultrasonidos
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);

   //PINs puente en H
  pinMode(pinIN1A, OUTPUT);
  pinMode(pinIN2A, OUTPUT);
  pinMode(pinIN1B, OUTPUT);
  pinMode(pinIN2B, OUTPUT);

  pixy.init();    //Inicialitza la camera
}

//*************************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio bucle, encarregada de sequenciar les funcions de tot el
programa y variar la velocitat dels robots depenent de la proximitat dels
objectes
//*************************************************************************//

void loop()
{
  pixy.ccc.getBlocks();
  if (pixy.ccc.numBlocks)   //Si detecta la camara entrara al loo de
moviments, sino el robot no fara res
   {
    camara();   //Entra a la funcio camara per saber cap a on s'ha de
moure
```

```
    obstaculos();   //Entra a la funcio obstaculos per fer la mediana de
la distancia que medeix l'ultraso
    correguir();    //Corregueix la direccio depenent de la distancia a
la que estigui el cotxe de davant
    movimiento_GPS();   //Entra a moviment GPS per accionar als motors
necessaris per el moviment
   }
}

//*********************************************************************//
// Nom de la funcio: movimiento_GPS()
// Funcio: Funcio encarregada d'activar els motors per direccionar el
robot amb velocitat variable.
//*********************************************************************//

void movimiento_GPS() {
 switch (salida) {
    case 1:   //Anar cap a dalt i cap a l'esquerre
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;

    case 2:   //Anar cap a dalt
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;

    case 3:   //Anar cap a dalt i cap a la dreta
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 4:   //Anar cap a baix i cap a la dreta
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 5:   //Anar cap a baix
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;

    case 6:   //Anar cap a baix i cap a l'esquerre
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;
```

```
    default:    //El estat default fa que els motors es quedin totalment
quiets
      digitalWrite(pinIN1A, LOW);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;
  }
}

//*********************************************************************//
// Nom de la funcio: correguir()
// Funcio: Funcio que varia la velicitat i la direccio dels motors
depenent de la distancia a la que estigui el cotxe de davant.
//*********************************************************************//

void correguir()
{
  if (med_ultra>50){    //Si la distancia es major a 50 cms
    salida=salida;    //La direccio sera la mateixa
    speed=speed;    //La velocitat sera la mateixa
  }
  else if ((20<med_ultra)&&(med_ultra<50)){   //Si la distancia esta
entre 20 i 50 cms
    salida=salida;   //La direccio sera la mateixa
    speed=50;    //La velocitat es reduira perque s'esta apropant al
cotxe de davant
  }
  else if ((med_ultra<20)&&(med_ultra>10)){   //Si la distancia esta
entre 10 i 20 cms
    salida=0;   //Es paren els motors
  }
  else {    //Si la distancia es menor a 10 cms
    salida=5;   //Marxa enrere perque el cotxe de davant esta massa aprop
o esta tirant marxa enrere
  }
}

//*********************************************************************//
// Nom de la funcio: obstaculos()
// Funcio: Funcio que calcula la distancia a la que esta el robot dels
objectes de devant (es fan medianas per tal de tindre valors mes
estables)
//*********************************************************************//

void obstaculos() {
  //Ultraso
  int i = 0;
  while (i <= 3) {
    digitalWrite(TRIGPIN, LOW);   //Pin de dispar a esta baix
    delayMicroseconds(5);
    digitalWrite(TRIGPIN, HIGH);    //Envia un estat alt per tal de saber
on es l'objecte
    delayMicroseconds(15);
    digitalWrite(TRIGPIN, LOW);   //Es torna a posar el valor del trigger
a 0
```

```
    distancia = pulseIn(ECHOPIN, HIGH);   //Llegueix el temps que ha
tardat en retornar el estat alt enviat anteriorment
    distancia = distancia / 58;   //Calcula la distancia del pols
(anteriorment es mesurava en temps)
    if (distancia>0){
      distancia_sonido[i] = distancia;    //S'emmagatzema 11 valors de
distancia
      i = i + 1;
    }
  }
  med_ultra = QuickMedian<int>::GetMedian(distancia_sonido,
3);    //Mitjancant la libreria es treu la mediana d'aquests 11 valors
mesurats anteriorment
}

//*********************************************************************//
// Nom de la funcio: camara()
// Funcio: Funcio encarregada de la desicio de moviment del robot
//*********************************************************************//

void camara()
{
    posX=pixy.ccc.blocks[0].m_x;
    if (posX<100) {   //Si la posicio que es detecta es inferior a 140,
el moviment ha de ser cap a endavant i cap a l'esquerre
      salida=1;
    }
    else if (posX>240){   //Si la posicio que es detecta es superior a
200, el moviment ha de ser cap a endavant i cap a la dreta
      salida=3;
    }
    else {    //Si esta entre 140 i 200 s'ha de moure recte
      salida=2;
    }
    speed=255;   //Velocitat en PWM
}
```

A3. Codi del tercer robot (Wall)

```
// Codi TFG Marc Guarnido Arquero i Sergio Ordiaz Bonet. Robot 3 "WallE"
par a automatizacio de processos forestals.

//Llibrerias
#include <TimerOne.h>   //Timer
#include <QuickMedianLib.h>   //Timer

//Pins Ultrasons dret
#define ECHOPIND 8    // Pin de lectura de la senyal de l'ultraso
#define TRIGPIND 7    // Pin de enviament de la senyal de l'ultraso

//Pins Ultrasons esquerre
#define ECHOPINI 3    // Pin de lectura de la senyal de l'ultraso
#define TRIGPINI 4    // Pin de enviament de la senyal de l'ultraso

//Pins pont en H
#define pinIN1A 6   //Pin per anar cap a darrere (acciona motor de
darrere)
```

```
#define pinIN2A 9   //Pin per anar cap endavant (acciona motor de
darrere)
#define pinIN1B 10    //Pin per anar cap a la dreta (acciona motor de
davant)
#define pinIN2B 11    //Pin per anar cap a l'esquerre (acciona motor de
davant)

//Variable PWM
int speed=250;

//Variable moviment
int salida;

//Variables ultraso
int med_ultra_d;
int med_ultra_i;
int distancia_sonido_d[3];
int distancia_d;
int distancia_sonido_i[3];
int distancia_i;

//**************************************************************************
*************************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//**************************************************************************
*************************************************************************//

void setup() {

  //PINs ultrasonidos
  pinMode(ECHOPIND, INPUT);
  pinMode(TRIGPIND, OUTPUT);
  pinMode(ECHOPINI, INPUT);
  pinMode(TRIGPINI, OUTPUT);


  //PINs puente en H
  pinMode(pinIN1A, OUTPUT);
  pinMode(pinIN2A, OUTPUT);
  pinMode(pinIN1B, OUTPUT);
  pinMode(pinIN2B, OUTPUT);
}

//**************************************************************************
*************************************************************************//
// Nom de la funcio: UltraDer()
// Funcio: Funcio que calcula la distancia a la que esta el robot dels
objectes de devant (es fan medianas per tal de tindre valors mes
estables) y per detectar cap a on s'esta movent el robot de davant
//**************************************************************************
*************************************************************************//

void UltraDer(){
  int i=0;

  //Lectura Ultraso
```

```
  while (i <= 3) {
    digitalWrite(TRIGPIND, LOW);   //Pin de dispar a esta baix
    delayMicroseconds(5);
    digitalWrite(TRIGPIND, HIGH);    //Envia un estat alt per tal de
saber on es l'objecte
    delayMicroseconds(15);
    digitalWrite(TRIGPIND, LOW);   //Es torna a posar el valor del
trigger a 0
    distancia_d= pulseIn(ECHOPIND, HIGH);   //Llegueix el temps que ha
tardat en retornar el estat alt enviat anteriorment
    distancia_d = distancia_d / 58;   //Calcula la distancia del pols
(anteriorment es mesurava en temps)
    if (distancia_d>0){
      distancia_sonido_d[i] = distancia_d;    //S'emmagatzema 11 valors
de distancia
      i = i + 1;
    }
  }
  med_ultra_d = QuickMedian<int>::GetMedian(distancia_sonido_d,
3);    //Mitjancant la libreria es treu la mediana d'aquests 11 valors
mesurats anteriorment
}

//*************************************************************************
**************************************************************************//
// Nom de la funcio: UltraIzq()
// Funcio: Funcio que calcula la distancia a la que esta el robot dels
objectes de devant (es fan medianas per tal de tindre valors mes
estables) y per detectar cap a on s'esta movent el robot de davant
//*************************************************************************
**************************************************************************//

void UltraIzq(){
    int i=0;

  //Lectura Ultrasonidos
  while (i <= 3) {
    digitalWrite(TRIGPINI, LOW);   //Pin de dispar a esta baix
    delayMicroseconds(5);
    digitalWrite(TRIGPINI, HIGH);    //Envia un estat alt per tal de
saber on es l'objecte
    delayMicroseconds(15);
    digitalWrite(TRIGPINI, LOW);   //Es torna a posar el valor del
trigger a 0
    distancia_i = pulseIn(ECHOPINI, HIGH);   //Llegueix el temps que ha
tardat en retornar el estat alt enviat anteriorment
    distancia_i = distancia_i / 58;   //Calcula la distancia del pols
(anteriorment es mesurava en temps)
    if (distancia_i>0){
      distancia_sonido_i[i] = distancia_i;    //S'emmagatzema 11 valors
de distancia
      i = i + 1;
    }
  }
  med_ultra_i = QuickMedian<int>::GetMedian(distancia_sonido_i,
3);    //Mitjancant la libreria es treu la mediana d'aquests 11 valors
mesurats anteriorment
}
```

```
//*********************************************************************
*********************************************************************//
// Nom de la funcio: Decision()
// Funcio: Funcio que decideix la velocitat y cap a on s'ha de moure el
robot depenent del que llegeixin els ultrasons
//*********************************************************************
*********************************************************************//

void Decision(){
  if (((med_ultra_d<=70) && (med_ultra_d>=15)) &&
((med_ultra_i<=70)&&(med_ultra_i>=15))){    //Si les dos mesures son
inferiors a a 30 cms, vol dir que ha de seguir recte
    salida=2;
  }
  else if ((med_ultra_d<=70) && (med_ultra_i>70)){    //Si l'ultraso de
l'esquerre detecte mes de 30 cms, significa que el robot de davant esta
girana la dreta perque s'allunya per el costat contrari
    salida=3;
  }
  else if ((med_ultra_d>70) && (med_ultra_i<=70)){    //Si l'ultraso de
la dreta detecte mes de 30 cms, significa que el robot de davant esta
girana l'esquerre perque s'allunya per el costat contrari
    salida=1;
  }
  else if ((med_ultra_d<=15) && (med_ultra_i<=15)){    //Si les mesures
son inferiors a 15 cms el cotxe ha de tirar marxa enrere perque esta
massa aprop el cotxe de davant
    salida=5;
  }
  else {    //Si no s'ha donat cap de les altres opcions parem el motor
porque el cotxe s'ha perdut
    salida=0;
  }
}

//*********************************************************************
*********************************************************************//
// Nom de la funcio: movimiento_GPS()
// Funcio: Funcio encarregada d'activar els motors per direccionar el
robot amb velocitat variable.
//*********************************************************************
*********************************************************************//

void movimiento_GPS(){
  switch (salida) {
    case 1:   //Anar cap a dalt i cap a l'esquerre
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;

    case 2:   //Anar cap a dalt
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
```

```
      break;

    case 3:   //Anar cap a dalt i cap a la dreta
      analogWrite(pinIN1A, speed);
      digitalWrite(pinIN2A, LOW);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 4:   //Anar cap a baix i cap a la dreta
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      analogWrite(pinIN1B, 255);
      digitalWrite(pinIN2B, LOW);
      break;

    case 5:   //Anar cap a baix
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;

    case 6:   //Anar cap a baix i cap a l'esquerre
      digitalWrite(pinIN1A, LOW);
      analogWrite(pinIN2A, speed);
      digitalWrite(pinIN1B, LOW);
      analogWrite(pinIN2B, 255);
      break;

    default:    //El estat default fa que els motors es quedin totalment
quiets
      digitalWrite(pinIN1A, LOW);
      digitalWrite(pinIN2A, LOW);
      digitalWrite(pinIN1B, LOW);
      digitalWrite(pinIN2B, LOW);
      break;
  }
}

//**********************************************************************
**********************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio bucle, encarregada de sequenciar les funcions de tot el
programa
//**********************************************************************
**********************************************************************//

void loop() {
  UltraDer();   //Llegeix l'ultraso dret
  UltraIzq();   //Llegeix l'ultrado esquerre
  Decision();   //Depenent de les lectures anteriors li asigna un valor o
un altre a salida
  movimiento_GPS();   //Acciona els motors depenent de la variable
salida, aportada per el GPS
}
```

A4. Codi del GPS associat al primer robot

```
#include "QuickMedianLib.h"
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <TimerOne.h>


SoftwareSerial mySerial(8, 7);
Adafruit_GPS GPS(&mySerial);

#define GPSECHO  true

double latitud_actual[3];
double longitud_actual[3];
double med_lat;
double med_long;
double latitud_objetivo;
double longitud_objetivo;
int i;
int j;
int dif_la;
int dif_ln;
int salida;

//**************************************************************************
*************************************************************************//
// Nom de la funcio: setup()
// Funcio: Funcio encarregada de inicialitzar les variables. Nomes
s'executa un cop.
//**************************************************************************
*************************************************************************//

void setup()
{
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(bumper);
  attachInterrupt(digitalPinToInterrupt(2),bumper,RISING);

  pinMode(5,OUTPUT);


  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);

  Serial.begin(115200);
  delay(5000);


  //Multiplicado por 10
  latitud_objetivo= 41309.9;
  longitud_objetivo= 2005.5;

  // Con esta declaracion evitamos un estado de indeterminacion en las
tres primeras lecturas del GPS (mediana)
  med_lat = latitud_objetivo;
```

```
  med_long = longitud_objetivo;

  // La velocidad de transmision, en baudios, que utiliza el GPS es de
9600
  GPS.begin(9600);

  // Activa el RMC (minimo recomendado) y el GGA (correcion de datos)
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

  // La frecuencia a la que se actualiza el dispositivo es de 1 Hz, no se
recomienda usar mas de 1 Hz, ya que podria ser que no diese tiempo a
ordenar datos y mostrarlos
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);

  // Solicita informacion sobre el estado de la antena
  GPS.sendCommand(PGCMD_ANTENNA);

  delay(1000);
}

//**************************************************************************
*************************************************************************//
// Nom de la funcio: bumper()
// Funcio: Funcio d'interrupcio per aturar el robot si colisiona contra
algun objecte.
//**************************************************************************
*************************************************************************//

void bumper (){
  digitalWrite(5, HIGH);
  delay(2000);
}

//**************************************************************************
*************************************************************************//
// Nom de la funcio: GPS()
// Funcio: Funcio encarregada de realitzar la lectura del GPS i extreure
un valor de la mediana de la longitud i la latitud.
//**************************************************************************
*************************************************************************//

uint32_t timer = millis();
void GPS()
{
  char c = GPS.read();


  // Es comproba si el GPS esta obtenint valors
  if (GPS.newNMEAreceived()) {
    if (!GPS.parse(GPS.lastNMEA()))
      return;
  }

  // Resetaer timer si super el valor del contador de millis
  if (timer > millis())  timer = millis();

  // Cada dos segundo se muestran las estadisticas actuales
  if (millis() - timer > 2000) {
```

```
    timer = millis(); // reset the timer

    if (GPS.fix) {

      // Se realiza una mediana de los valores obtenidos por el GPS para
qur sean mas fiables (se cogen 3 valores para hacer la media)
      latitud_actual[i]=GPS.latitude*10;
      i=i+1;
      if (i>=3){
        i=0;
        med_lat=QuickMedian<double>::GetMedian(latitud_actual, 3);
        Serial.println(med_lat);
      }

      longitud_actual[j]=GPS.longitude*10;
      j=j+1;
      if (j>=3){
        j=0;
        med_long=QuickMedian<double>::GetMedian(longitud_actual, 3);
        Serial.println(med_long);
        Serial.println(salida);
      }
    }
  }
}

//**********************************************************************
**********************************************************************//
// Nom de la funcio: decision_salida()
// Funcio: Funcio encarregada de donar-li valor a la variable salida
dependent de la posició en la que es trobi el robot.
//**********************************************************************
**********************************************************************//

void decision_salida();{

  dif_la=latitud_objetivo-med_lat;
  dif_ln=longitud_objetivo-med_long;


  // Movimiento hacia adelante y hacia la izquierda
  if ((dif_la < -0.1) and (dif_ln > 0.1)){
    digitalWrite(1,LOW);
    digitalWrite(2,LOW);
    digitalWrite(3,HIGH);
    salida=1;   //Salida digital 001, caso 1
  }

  // Movimiento hacia adeltane
  if ((dif_la > -0.1) and (dif_la<0.1) and (dif_ln > 0.1)){
    digitalWrite(1,LOW);
    digitalWrite(2,HIGH);
    digitalWrite(3,LOW);
    salida=2;   //Salida digital 010, caso 2
  }

  // Movimiento hacia adeltane y hacia la derecha
  if ((dif_la > 0.1) and (dif_ln > 0.1)){
```

```
    digitalWrite(1,LOW);
    digitalWrite(2,HIGH);
    digitalWrite(3,HIGH);
    salida=3;   //Salida digital 011, caso 3
  }

  // Moviento hacia atras y hacia la derecha
  if ((dif_la > 0.1) and (dif_ln < -0.1)){
    digitalWrite(1,HIGH);
    digitalWrite(2,LOW);
    digitalWrite(3,LOW);
    salida=4;   //Salida digital 100, caso 4
  }

  // Movimiento hacia atras

  if ((dif_la > -0.1) and (dif_la<0.1) and (dif_ln < -0.1)){
    digitalWrite(1,HIGH);
    digitalWrite(2,LOW);
    digitalWrite(3,HIGH);
    salida=5;   //Salida digital 101, caso 5
  }

  //Movimiento hacia atras y hacia la izquierda
  if ((dif_la < -0.1) and (dif_ln < -0.1)){
    digitalWrite(1,HIGH);
    digitalWrite(2,HIGH);
    digitalWrite(3,LOW);
    salida=6;   //Salida digital 110, caso 6
  }
}

//**********************************************************************
**********************************************************************//
// Nom de la funcio: loop()
// Funcio: Funcio bucle, encarregada de sequenciar les funcions de tot el
programa y variar el valor de salida
//**********************************************************************
**********************************************************************//

void loop(){
  GPS();
  decisión_salida();
}
```